

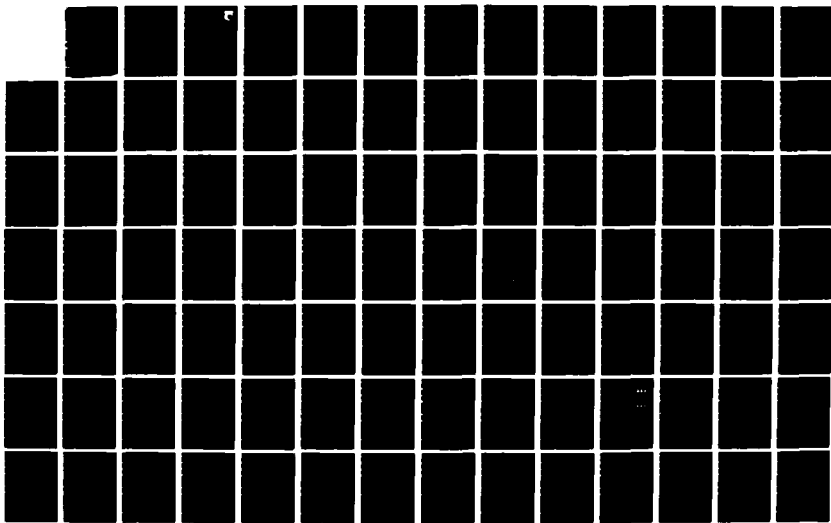
AD-A188 529

THE ACOUSTIC-MODELING PROBLEM IN AUTOMATIC SPEECH  
RECOGNITION(U) CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT  
OF COMPUTER SCIENCE P F BROWN DEC 87 CMU-CS-87-125  
AFWAL-TR-87-1161 F33615-84-K-1528 F/G 25/4

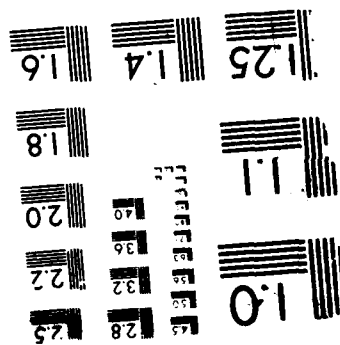
1/2

UNCLASSIFIED

NL



W. H. COOPY RESOLUTION TEST CHART



PHOTOGRAPH THIS SHEET

AD-A188 529

DTIC ACCESSION NUMBER

LEVEL

INVENTORY

AFWAL-TR-87-1161

DOCUMENT IDENTIFICATION

Dec 1987

DISTRIBUTION STATEMENT

ACCESSION FOR

NTIS GRA&I ☒

DTIC TAB ☐

UNANNOUNCED ☐

JUSTIFICATION

BY

DISTRIBUTION /

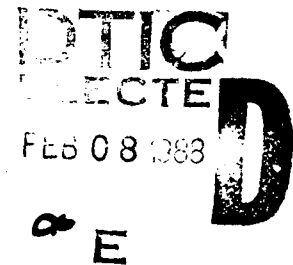
AVAILABILITY CODES

DIST

AVAIL AND/OR SPECIAL

A-1

DISTRIBUTION STAMP



DATE ACCESSIONED

DATE RETURNED

88 2 05 107

DATE RECEIVED IN DTIC

REGISTERED OR CERTIFIED NO.

PHOTOGRAPH THIS SHEET AND RETURN TO DTIC-DDAC

AD-A188 529

AFWAL-TR-87-1161

THE ACOUSTIC-MODELING PROBLEM IN AUTOMATIC SPEECH  
RECOGNITION

Peter F. Brown

Carnegie-Mellon University  
Computer Science Department  
Pittsburgh, PA 15213-3890

December 1987

Interim



Approved for Public Release; Distribution is Unlimited

AVIONICS LABORATORY  
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES  
AIR FORCE SYSTEMS COMMAND  
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6543

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

## REPORT DOCUMENTATION PAGE

Form Approved  
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified		1b. RESTRICTIVE MARKINGS	
2a. SECURITY CLASSIFICATION AUTHORITY		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.	
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE		5. MONITORING ORGANIZATION REPORT NUMBER(S) AFWAL-TR-87-1161	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) CMU-CS-87-125		7a. NAME OF MONITORING ORGANIZATION Air Force Wright Aeronautical Laboratories AFWAL/AAAT-3	
6a. NAME OF PERFORMING ORGANIZATION Carnegie-Mellon University	6b. OFFICE SYMBOL (If applicable)	7b. ADDRESS (City, State, and ZIP Code) Wright-Patterson AFB OH 45433-6543	
6c. ADDRESS (City, State, and ZIP Code) Computer Science Dept Pittsburgh PA 15213-3890		9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F33615-84-K-1520	
8a. NAME OF FUNDING/SPONSORING ORGANIZATION	8b. OFFICE SYMBOL (If applicable)	10. SOURCE OF FUNDING NUMBERS	
5c. ADDRESS (City, State, and ZIP Code)		PROGRAM ELEMENT NO. 61101E	PROJECT NO. 4976
		TASK NO. 00	WORK UNIT ACCESSION NO. 01
11. TITLE (Include Security Classification) The Acoustic-Modeling Problem In Automatic Speech Recognition			
12. PERSONAL AUTHOR(S) Peter F. Brown			
13a. TYPE OF REPORT Interim	13b. TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Year, Month, Day) 1987 December	15. PAGE COUNT 126
16. SUPPLEMENTARY NOTATION			
17. COSATI CODES		18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)	
FIELD	GROUP	SUB-GROUP	
19. ABSTRACT (Continue on reverse if necessary and identify by block number)			
<p>This thesis examines the acoustic-modeling problem in automatic speech recognition from an information-theoretic point of view. This problem is to design a speech-recognition system which can extract from the speech waveform as much information as possible about the corresponding word sequence. The information extraction process is broken down into two steps: a signal-processing step which converts a speech waveform into a sequence of information-bearing acoustic feature vectors, and a step which models such a sequence. This thesis is primarily concerned with the use of hidden Markov models to model sequences of</p>			
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT <input checked="" type="checkbox"/> UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS RPT. <input type="checkbox"/> DTIC USERS		21. ABSTRACT SECURITY CLASSIFICATION Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL Chahira M. Hopper		22b. TELEPHONE (Include Area Code) (513) 255-7865	22c. OFFICE SYMBOL AFWAL/AAAT-3

DD Form 1473, JUN 86

Previous editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE  
Unclassified

UNCLASSIFIED

Block 19 (Continued)

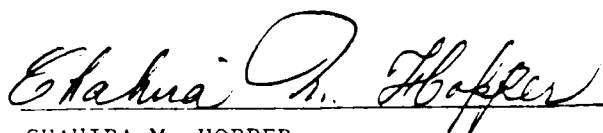
feature vectors which lie in a continuous space such as  $R^N$ . It explores the trade-off between packing a lot of information into such sequences and being able to model them accurately. The difficulty of developing accurate models of continuous-parameter sequences is addressed by investigating a method of parameter estimation which is specifically designed to cope with inaccurate modeling assumptions.

## NOTICE

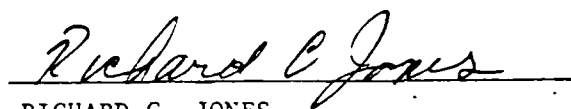
When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility or any obligation whatsoever. The fact that the Government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise in any manner construed, as licensing the holder, or any other person or corporation; or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Office of Public Affairs (ASD/PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.



CHAHIRA M. HOPPER  
Project Engineer



RICHARD C. JONES  
Ch, Advanced Systems Research Gp  
Information Processing Technology Br

FOR THE COMMANDER



EDWARD L. GLIATTI  
Ch, Information Processing Technology Br  
Systems Avionics Div

If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify AFWAL/AAAT, Wright-Patterson AFB, OH 45433-6543 to help us maintain a current mailing list.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

## Acknowledgements

The research described in this thesis was carried out in conjunction with other members of the IBM Continuous Speech Recognition group. I would like to thank Amir Averbuch, Lalit Bahl, Raimo Bakis, Peter de Souza, Gopalakrishnan, Fred Jelinek, Bob Mercer, Arthur Nadas, David Nahamoo, Michael Picheny, and Dirk Van Compernelle for their invaluable assistance. I worked particularly closely with Lalit, Peter, and Bob. One of the key concepts in this thesis, maximum mutual information estimation, evolved directly out of conversations between the four of us. Time and time again, I would come up with some idea, and then realise that it was just something that Bob had urged me to try out months before. It was as if, step by step, I was uncovering some master plan of his. Peter and I have adjacent offices at IBM. As soon as either of us has an idea or interesting result, we run next door to discuss it with the other. There is no telling to what extent the work in this thesis derives from these discussions. I am especially indebted to Peter as well as to Gopalakrishnan and Kai-Fu Lee for the hundreds of problems they uncovered while proofreading this thesis.

I would also like to thank Jim Baker, who first introduced me to the hidden-Markov approach to automatic speech recognition, and Geoff Hinton, my advisor at Carnegie-Mellon. From Geoff I learned that research in artificial intelligence can be carried out by solving well defined optimisation problems using models with explicit assumptions.

In addition to Geoff Hinton and Bob Mercer, I am grateful to Raj Reddy and Richard Stern for serving on my thesis committee.

I would like to thank Peggy Hamburg for the many weekends she spent hanging around my apartment, keeping me company, while I was pounding away at the keyboard writing this thesis.

Finally, I thank my parents for their support and for all that they have taught me during the past thirty two years.



# Table of Contents

<b>1. Introduction</b>	1
<b>2. An Information Theoretic Approach</b>	4
2.1 Entropy and Mutual Information	4
2.2 Maximum Likelihood Estimation	9
2.3 Maximum Mutual Information Estimation	11
<b>3. Hidden Markov Models</b>	17
3.1 Basics	17
3.2 Modifications to the Basic Model	21
3.3 The Forward-Backward Algorithm	24
3.4 Maximum Mutual Information Estimation	29
<b>4. The IBM Experimental Speech Recognizer</b>	34
<b>5. Continuous Parameters</b>	37
5.1 The Quantization Problem	37
5.2 Output Densities	39
5.3 Mixtures	44
<b>6. Parameter Reduction</b>	47
6.1 The Parameter Reduction Problem	47
6.2 Linear Discriminants	48
6.3 Application to Speech Recognition	51
<b>7. Adjoining Acoustic Parameter Vectors</b>	54
<b>8. Conditional Models</b>	56
8.1 The Correlation Problem	56
8.2 The Continuous Case	57
8.3 The Discrete Case	59
8.4 Run-Length Adjustment	63
<b>9. An LPC-Based Waveform Model</b>	67
9.1 The Framing Problem	67
9.2 Linear Predictive Models	69
<b>10. Implementation</b>	75
10.1 Thresholding the Forward-Backward Computation	75
10.2 Representing Probabilities by Their Logarithms	78
10.3 Maximum Mutual Information Estimation	80
<b>11. E-Set Experiments</b>	83
11.1 Previous Results	83
11.2 Data	85
11.3 Models and Methods	87
11.4 Resources	90
11.5 Results	92
<b>12. Summary and Conclusion</b>	100
<b>13. Appendices</b>	104
13.1 Differentiation Rules	104
13.2 Diagonal Gaussian Formulas	107
13.3 Full-Covariance Gaussian Formulas	108
13.4 Richter Type Gaussian Formulas	109
13.5 Conditional Gaussian Formulas	111
13.6 LPC Formulas	114
<b>14. References</b>	115

## Figures and Tables

Table 2.1	Joint Probabilities of Words and Acoustic Signals . . . . .	14
Table 2.2	MLE vs. MMIE, Assuming True Language Model is Known . . . . .	14
Table 2.3	Joint Probabilities Computed with Incorrect Language Model . . . . .	15
Table 2.4	MLE vs. MMIE, Assuming Incorrect Language Model . . . . .	15
Figure 3.1	Sample Hidden Markov Model Diagram . . . . .	23
Figure 5.1	Simple Model for <i>the</i> . . . . .	41
Figure 5.2	Complicated Model for <i>the</i> . . . . .	41
Figure 5.3	Richter (solid line) vs. Gaussian Density (dashed line) . . . . .	43
Figure 5.4	Richter (solid line) vs. Gaussian Log Density (dashed line) . . . . .	43
Figure 5.5	Simple Model with Mixture Problem for MLE . . . . .	44
Figure 6.1	Two Gaussian Classes . . . . .	49
Figure 7.1	Time-Derivative Modeling by Machine Alteration . . . . .	55
Table 8.1	Average Ratios of Observed to Predicted Run-Length . . . . .	64
Figure 9.1	Initial Section of <i>v</i> Waveform . . . . .	71
Figure 9.2	Scores from Basic LPC Model . . . . .	71
Figure 9.3	Scores from LPC Model with Pitch Tracker . . . . .	71
Figure 9.4	Waveform Phone Machine . . . . .	73
Table 9.1	Phonetic Spellings for Waveform Experiment . . . . .	73
Table 10.1	$\psi$ Lookup Table Sizes . . . . .	79
Table 11.1	Phonetic Categories for E-Set Experiments . . . . .	86
Table 11.2	Phonetic Spellings for E-Set Experiment . . . . .	86
Table 11.3	Signal-to-Noise Ratios in Decibels . . . . .	87
Figure 11.1	Onset Phone Machine . . . . .	88
Figure 11.2	Normal Phone Machine . . . . .	88
Table 11.4	Approximate IBM-3090 CPU Hours for Experiments . . . . .	91
Figure 11.3	Number of Parameters vs. Performance on Test Data . . . . .	95
Figure 11.4	Number of Parameters vs. Mutual Information on Test Data . . . . .	96
Table 11.5	22-Parameter Results . . . . .	96
Table 11.6	12-Parameter Results . . . . .	97
Table 11.7	Abbreviations used in Table 11.7 . . . . .	98
Table 11.8	E-Set Results (See previous page for symbol definitions.) . . . . .	99

# 1. Introduction

A speech recognition system is a device which takes as input the acoustic waveform produced by the speaker using the device and produces as output the sequence of words corresponding to the input waveform. Speech recognition is a difficult task because it is uncertain what words will be spoken to the system, and because it is uncertain what acoustic waveform will be produced given that certain words have been spoken.

A speaker has a choice of what topic to discuss. For example, he might be dictating an article on algebraic geometry, or be talking to someone at an airline reservation counter. Once the general topic has been resolved there is still uncertainty in what message the speaker will convey. He may inquire about flights to Cleveland, or he may cancel his reservation to Kampala. In a natural language like English there are many different syntactic forms which can be used to communicate the same message. Once the topic, message, and syntactic form have been resolved a speaker still has the choice of what words to use. These examples of semantic, syntactic, and lexical uncertainty all contribute to the uncertainty of what words will be spoken to a speech recognition system. A formal measure of this uncertainty is the entropy of the recognizer's language.

The acoustic uncertainty also has many components. There is the uncertainty of accent in a speaker-independent system. The vowels of speakers from different geographic locations and from different ethnic backgrounds differ to a large extent. There is uncertainty as to what the general quality of a speaker's voice will be. This is what separates a song sung by Chuck Berry from one sung by Rudy Vallee. A given speaker may speak loudly or softly. He may speak quickly or slowly. He may be relaxed or under stress. His mouth may be close to or far from the microphone. He may be in the cockpit of a jet fighter, or in a soundproof room. He may have a cold. All of this uncertainty must be taken into account.

There are two sources of information which can be used to recognize a word, the words which have been spoken previously, and the acoustic signal. A speech recogniser uses a *language model* to extract information from the previous words spoken, and a family of *acoustic models* to extract information from the acoustic signal.

By extracting information from previous words a recogniser attempts to minimise the uncertainty, prior to examining the acoustic signal, of what the current word will be. On average, this uncertainty is bounded from below by the per-word entropy of the recogniser's language. If the recogniser knew the true model for its language, then it would be able to achieve this minimum uncertainty. If the recogniser's model is not correct, its uncertainty

will be greater. Speech recognition systems that use an artificial grammar do so in order to set this uncertainty by fiat, thereby ensuring that their tasks will not be too difficult.

After information has been extracted from the previous words spoken, any remaining uncertainty must be minimized by extracting information from the acoustic signal. As we shall discuss in detail, a formal measure of the maximum amount of information about a word that can be gleaned from an acoustic signal is the mutual information between the signal and the word. But, assuming the true language model is known, this upper bound can be obtained only by discovering the true models which specify with what probability a particular acoustic signal will be generated by a speaker pronouncing a particular word. If the acoustic models used by the recognizer are inaccurate, it will not be able to extract as much information.

The performance of a speech recognition system depends on the extent to which it can reduce the uncertainty of what a word is given an acoustic signal and a sequence of previous words. A system's performance, therefore, depends on the accuracy of both the system's language model and its acoustic models. Consequently, research in automatic speech recognition is divided into two basic areas, language modeling and acoustic modeling. In this thesis, I shall be concerned exclusively with acoustic modeling.

In the early days of speech recognition, acoustic models were created by linguistic experts, who attempted to express their knowledge of acoustic-phonetic rules in programs which analyzed an input speech signal and produced as output a sequence of phonemes [Dixon 76]. It was thought to be, and no doubt is, a simple matter to decode a word sequence from a sequence of phonemes. It turns out, however, to be a very difficult job to determine an accurate phoneme sequence from a speech signal. Although human experts, such as Victor Zue [Cole 80], certainly do exist, it has proven extremely difficult to formalize their knowledge in such a way that it can be incorporated into a computer program.

During the past fifteen years, an alternative, statistical, approach to acoustic modeling has been developed. This approach confronts the inability of human experts to completely formalize their knowledge by employing statistical methods that are capable of learning acoustic-phonetic knowledge from samples of speech. The predominant class of models that has been used in this statistical approach is referred to as *hidden Markov models*.

Hidden Markov models were first applied to automatic speech recognition in the early 1970's [Baker 75] [Bakis 76]. During the last decade, the hidden Markov modeling approach has become increasingly popular and successful. Today, it dominates the field. The reason for the success of this approach is that hidden Markov models are complex enough to begin

to represent speech as a sequence of acoustic-phonetic events of variable duration, and yet simple enough that their parameters can be estimated automatically from sample speech.

The method of parameter estimation that has been used with hidden Markov models is maximum likelihood estimation (MLE). There are many very important properties of MLE, but most of them stem from an implicit assumption of model correctness. The justifications for using MLE to estimate the mean and variance of a Gaussian distribution, for example, presume that the sample data has indeed been generated by a Gaussian. Currently we do not know of a *correct* model for speech. We can be almost certain that the assumptions made by any existing model of speech are not totally correct. We must, therefore, question the use of MLE. The objective in MLE is to do as good a job as possible of deriving the *true* model parameters. But if the assumptions implicit in the class of models for which parameter estimates are being derived are not true of the sample data, then it is not clear what true parameters would be. If there are no true parameters, then it is also no longer clear just what the rationale for MLE is. In this thesis, I shall present an alternative method of parameter estimation, *maximum mutual information estimation* (MMIE), which does not derive its rationale from presumed model correctness. In particular, the objective of MMIE is to find a set of parameters in such a way that the resultant acoustic models allow the system to derive from the acoustic signal as much information as possible about the corresponding word sequence.

We will see that MMIE can lead to a performance improvement over MLE. It is important to understand, however, that assuming the true language model is known, the upper bound on the acoustic information available in a system based on incorrect modeling assumptions is necessarily less than the mutual information between the acoustic signal and the word sequence. Only a system based on accurate assumptions about speech can achieve this mutual-information limit. Thus it is still of primary importance to improve the accuracy of the modeling assumptions made about speech. As these assumptions become more accurate, the performance advantage of estimates derived by MMIE over those derived by MLE will disappear.

The speech recognition group at IBM's Watson Research center has been using hidden Markov models since its inception in 1972. The IBM 20,000 word natural-language speech recogniser is generally accepted as being the most advanced speech-recognition system in the world today. In this thesis, I describe my attempts improve the performance of this system both by improving the acoustic-modeling assumptions in the system, and by using MMIE to reduce the impact of modeling inaccuracies which remain.

## 2. An Information Theoretic Approach

### 2.1 Entropy and Mutual Information

An intuitively plausible measure of the uncertainty in a random event  $X$  is the average number of bits necessary to specify the outcome of  $X$  under an optimal encoding scheme. By the fundamental theorem of information theory, also known as the noiseless coding theorem, this measure is the *entropy* of  $X$  [Shannon 48]:

$$H(X) \triangleq - \sum_x \Pr(X = x) \log \Pr(X = x). \quad (2.1)$$

Similarly, a formal measure of the uncertainty in a random event  $X$  given the outcome of a random event  $Y$  is the *conditional entropy* of  $X$  given  $Y$ :

$$H(X | Y) \triangleq - \sum_{x,y} \Pr(X = x, Y = y) \log \Pr(X = x | Y = y). \quad (2.2)$$

An intuitively plausible measure of the average amount of information provided by the random event  $Y$  about the random event  $X$  is the average difference between the number of bits it takes to specify the outcome of  $X$  when the outcome of  $Y$  is not known and when the outcome of  $Y$  is known. This is just the difference in the entropy of  $X$  and the conditional entropy of  $X$  given  $Y$ :

$$\begin{aligned} I(X; Y) &\triangleq H(X) - H(X | Y) \\ &= \sum_{x,y} \Pr(X = x, Y = y) \log \frac{\Pr(X = x, Y = y)}{\Pr(X = x) \Pr(Y = y)}. \end{aligned} \quad (2.3)$$

Since  $I(X; Y) = I(Y; X)$ ,  $I(X; Y)$  is referred to as the *average mutual information* between  $X$  and  $Y$ .

Let  $W$  be a random variable over sequences of words. Let  $Y$  be a random variable over sequences of acoustic information. On average, the uncertainty of a word sequence given a sequence of acoustic information is the conditional entropy of  $W$  given  $Y$ :

$$H(W | Y) = H(W) - I(W; Y). \quad (2.4)$$

In order to understand  $H(W | Y)$  more clearly, consider the following situation. Jack wants to communicate a message in a sequence of words,  $w$ , to Jill. He does this by

speaking the words, thereby producing the acoustic signal  $y$ . He knows that Jill hears  $y$ , but wants to make sure that Jill gets the message in  $w$ . To do this he sends her an additional string of bits,  $b$ , of length  $|b|$ . The fundamental theorem of information theory tells us that  $|b| \geq H(W | Y)$ . To achieve this average minimum length of  $b$  Jack uses his knowledge of the conditional probability of a word sequence given an acoustic signal, and he assumes that Jill has the same knowledge. The fundamental theorem tells us, specifically, that Jack's optimal encoding scheme uses  $-\log \Pr(W = w | Y = y)$  bits to tell Jill about the particular string,  $w$ , given that she has received the particular acoustic signal  $y$ .

Suppose that Jack and Jill do not know  $\Pr(W = w | Y = y)$ , but instead use a possibly incorrect probability distribution. Suppose they compute their probabilities with a model,  $m$ , and think that  $\Pr(W = w | Y = y)$  is actually  $\Pr_m(W = w | Y = y)$ . Let  $p(y)$ ,  $p(w, y)$ ,  $p(w | y)$ , and  $q(w | y)$  be abbreviations for  $\Pr(Y = y)$ ,  $\Pr(W = w, Y = y)$ ,  $\Pr(W = w | Y = y)$ , and  $\Pr_m(W = w | Y = y)$ , respectively. After Jill has received  $y$ , Jack will use  $-\log q(w | y)$  bits to tell Jill that the message is  $w$ . On average, the number of bits Jack will have to send her will be

$$\begin{aligned}
 H_m(W | Y) &= - \sum_{w,y} p(w, y) \log q(w | y) \\
 &= - \sum_{w,y} p(w, y) \log \left( \frac{q(w | y)}{p(w | y)} \right) - \sum_{w,y} p(w, y) \log p(w | y) \\
 &= - \sum_{w,y} p(w, y) \log \left( \frac{q(w | y)}{p(w | y)} \right) + H(W | Y) \\
 &\geq - \sum_{w,y} p(w, y) \left( \frac{q(w | y)}{p(w | y)} - 1 \right) + H(W | Y), \quad \text{since } \log z \leq z - 1 \\
 &\geq - \sum_{w,y} p(y) q(w | y) + 1 + H(W | Y) \\
 &\geq H(W | Y).
 \end{aligned} \tag{2.5}$$

Furthermore, because  $z - 1 = \log z$  only if  $z = 0$ ,  $H_m(W | Y) > H(W | Y)$  unless for all  $w$  and  $y$ ,  $q(w | y) = p(w | y)$ . † In other words, if Jack and Jill do not know the probabilities accurately, it will, on average, require more bits to make sure that Jill gets the message in  $w$ . Since  $\left[ \left( \frac{q(w | y)}{p(w | y)} - 1 \right) - \log \left( \frac{q(w | y)}{p(w | y)} \right) \right]$  increases monotonically with  $|q(w | y) - p(w | y)|$ , the extent to which the average length of  $b$  is greater than  $H(W | Y)$  depends on how far the  $q$ 's are from the true probabilities. If Jack and Jill want to keep their messages short, they should use as accurate an estimate as possible of the probability distribution of word strings given acoustic signals.

† This inequality is just a special case of what is known as Jensen's inequality.

Because Jill cannot do a perfect job of decoding  $w$  from  $y$ , she needs the additional information in  $b$ . The amount of additional information she requires to be sure of the identity of  $w$  is  $|b|$  bits. The average length of  $b$ ,  $H_m(W | Y)$ , is a measure of her average uncertainty of a word string  $w$  after having received the acoustic data  $y$ . The better Jill's model of the probability distribution of words given speech, the more certain she will be of the identity of the word string which has been spoken to her.

Let us now replace Jill by a speech recognition system. Suppose, as is normally the case, that the speech recognition system receives no additional information. Following Nadas [Nadas 83], the recognizer can be thought of as a decoding function  $f: y \rightarrow w$ . The probability that it will decode a word sequence correctly is

$$v(f) \triangleq \sum_y \Pr(W = f(y), Y = y). \quad (2.6)$$

Since

$$\sum_y \Pr(W = f(y), Y = y) \leq \sum_y \max_w \Pr(W = w, Y = y), \quad (2.7)$$

an optimal recognizer will choose  $f(y)$  such that

$$\Pr(W = f(y) | Y = y) = \max_w \Pr(W = w | Y = y). \quad (2.8)$$

Because a real-life recognizer does not actually know the true probabilities, but instead estimates  $\Pr(W = w | Y = y)$  by  $\Pr_m(W = w | Y = y)$ , it will choose  $f(y)$  so that

$$\Pr_m(W = f(y) | Y = y) = \max_w \Pr_m(W = w | Y = y). \quad (2.9)$$

To increase the chance that the recognizer will guess a word correctly, it should use a model in which  $\Pr_m(W = w | Y = y)$  is large when the speaker is likely to have spoken the word sequence  $w$  while generating the speech  $y$ , that is, when  $\Pr(W = w, Y = y)$  is large. We can see from equation (2.5) that such models will tend to make  $H_m(W | Y)$  small, since the logarithm is a monotonically increasing function. Decreasing the recognizer's average uncertainty  $H_m(W | Y)$  will thus tend to increase its probability of guessing a correct word sequence.

Traditionally, the model  $m$  used in a speech recognition system is partitioned into two components, a language model, and an acoustic model. In order to see how these two components determine the recognizer's output  $f(y)$  for a given input  $y$ , we can use Bayes' rule to reformulate equation (2.9) as

$$\frac{\Pr_m(Y = y | W = f(y)) \Pr_m(W = f(y))}{\Pr_m(Y = y)} = \max_w \frac{\Pr_m(Y = y | W = w) \Pr_m(W = w)}{\Pr_m(Y = y)}. \quad (2.10)$$



Here  $\text{Pr}_m(W = w)$  is the prior probability that a word sequence  $w$  will be spoken, and is computed with the language model.  $\text{Pr}_m(Y = y | W = w)$  is the conditional probability that the speaker will produce the acoustic signal  $y$  given that he will speak the word sequence  $w$ , and is computed with the acoustic model.  $\text{Pr}_m(Y = y)$  is independent of any particular  $w$  and therefore does not effect the system's behavior.

We would like to choose the model,  $m$ , to minimize  $H_m(W | Y)$ . Analogous to equation (2.4) we have

$$H_m(W | Y) = H_m(W) - I_m(W; Y), \quad (2.11)$$

in which

$$H_m(W) = - \sum_w \text{Pr}(W = w) \log \text{Pr}_m(W = w), \quad (2.12)$$

and

$$I_m(W; Y) = \sum_{w, y} \text{Pr}(W = w, Y = y) \log \frac{\text{Pr}_m(W = w, Y = y)}{\text{Pr}_m(W = w) \text{Pr}_m(Y = y)}. \quad (2.13)$$

Although in certain tasks it is possible to search for a model with the goal of minimizing  $H(W | Y)$  directly [Hinton 86], in speech recognition, the problem of minimizing  $H_m(W | Y)$  is normally divided into two separate subproblems: finding a language model which minimises the first term  $H_m(W)$ , and finding an acoustic model which maximises  $I_m(W; Y)$ .

Let us first briefly consider the language model. By a derivation similar to (2.5), it is easily seen that  $H_m(W)$  is bounded from below by  $H(W)$  and can be minimised by choosing  $\text{Pr}_m(W = w)$  to be as close as possible to  $\text{Pr}(W = w)$ , for each sequence of words  $w$ . In a system with an artificial grammar, the  $\text{Pr}(W = w)$ 's are known and  $H_m(W)$  can, in fact, achieve its lower bound if the system simply uses these probabilities. In a natural language system, the true probabilities are not known.  $H_m(W)$  cannot be computed outright, but must, instead, be estimated from a large sample of text, and the language model should be designed to minimise this estimate. In this thesis, I will not be further concerned with the language modeling problem. I will just assume that the system's language model, accurate or not, is given.

Once a language model has been chosen, the acoustic modeling task is to find an acoustic model which maximises  $I_m(W; Y)$ , which is just a measure of the average number of bits of information about a word string  $w$ , a recogniser is able to extract from an acoustic signal  $y$  using the model  $m$ . As in the language modeling task, the true probabilities are not known, and  $I_m(W; Y)$  cannot be computed outright, but must, instead, be estimated from

sample data. In this case the sample data consists of a sample of text and of a sample of speech. The speech,  $y$ , is the acoustic signal which was produced when the word sequence,  $w$ , was spoken. The goal of acoustic modeling is to maximize the estimate of  $I_m(W; Y)$  on whatever sample data have been collected, in an effort to derive a model with which the system can extract as much information as possible from a test acoustic signal about the corresponding test word sequence.

For our purposes, an acoustic model is a probability distribution which models the phonological and acoustic-phonetic variations found in the speech presented to the recogniser. It is impossible for a human being to write down an accurate and complete acoustic model. For this reason, when designing an acoustic model, one normally develops a family of possible distributions, parameterized by a vector of parameters,  $\theta$ , which is estimated from sample speech. For example, one might decide that the conditional distribution of an acoustic signal given a word is a member of the family of multivariate Gaussians. Each such Gaussian distribution would be parameterized by a mean and covariance matrix. The model,  $m$ , which we have been discussing is specified by specifying the distribution family, and by specifying the set of parameters which identify a particular distribution within the family. Deciding on the family is the art in speech recognition; deciding on the parameters is the science.

The issue of whether to account for an acoustic phenomenon in the constraints inherent in the distribution family, or to account for it in the particular values of the model parameters is an important one. If the family is over-restrictive, it may be incorrect. If it is under-restrictive, there may be too many model parameters to estimate reliably from a reasonably sized sample of speech. Ideally, one would like to use a family of distributions which is based on true assumptions about speech and has as few free parameters as possible. In practice, the best one can hope for is a compromise in which some model assumptions are inaccurate but not wildly inaccurate, and in which parameter estimates may be off, but not way off. In subsequent chapters, we will consider a particular class of distribution families in detail. In the remainder of this chapter, we will consider the problem of estimating a vector of parameters,  $\theta$ , from a training sample  $(w, y)$ . A parameter estimation method can be thought of as a function  $g : (w, y) \rightarrow \theta$ . In this thesis, we will compare two methods of estimating the parameters of an acoustic model: maximum likelihood estimation and maximum mutual information estimation.

## 2.2 Maximum Likelihood Estimation

In maximum likelihood estimation a parameter vector,  $\hat{\theta} = g(\mathbf{w}, \mathbf{y})$ , is derived so that

$$\Pr_{\hat{\theta}}(\mathbf{W} = \mathbf{w}, \mathbf{Y} = \mathbf{y}) = \max_{\theta} \Pr_{\theta}(\mathbf{W} = \mathbf{w}, \mathbf{Y} = \mathbf{y}), \quad (2.14)$$

where  $\Pr_{\theta}(\mathbf{W} = \mathbf{w}, \mathbf{Y} = \mathbf{y})$  is the probability that the model parameterized by  $\theta$  in the family under consideration will generate the sample  $(\mathbf{w}, \mathbf{y})$ . By factoring  $\Pr_{\theta}(\mathbf{W} = \mathbf{w}, \mathbf{Y} = \mathbf{y})$  as

$$\Pr_{\theta}(\mathbf{W} = \mathbf{w}, \mathbf{Y} = \mathbf{y}) = \Pr_{\theta}(\mathbf{Y} = \mathbf{y} | \mathbf{W} = \mathbf{w}) \Pr_{\theta}(\mathbf{W} = \mathbf{w}), \quad (2.15)$$

we can see that acoustic model parameters and language model parameters can be estimated separately by choosing the language model parameters to maximize  $\Pr_{\theta}(\mathbf{W} = \mathbf{w})$  and by choosing the acoustic model parameters to maximize  $\Pr_{\theta}(\mathbf{Y} = \mathbf{y} | \mathbf{W} = \mathbf{w})$ .

In the previous section, we saw that if the true acoustic and language models were known, we could construct an optimal speech recognition system. This suggests that the closer the estimator  $g(\mathbf{w}, \mathbf{y})$  is to the true parameter vector,  $\tilde{\theta}$ , the closer the performance of the recognizer will be to optimal performance. Arthur Nadas [Nadas 83] has shown that this argument, when fleshed out formally with the appropriate assumptions, leads one to maximum likelihood estimation. Because maximum likelihood estimation is by far and away the most common method of parameter estimation in pattern recognition, we will briefly review the primary argument for its use.

An estimator,  $g(\mathbf{w}, \mathbf{y})$ , is a function of samples of the random variables  $\mathbf{W}$  and  $\mathbf{Y}$ , and is therefore itself a random variable, with a distribution determined by the distributions of  $\mathbf{W}$  and  $\mathbf{Y}$ . Let  $\tilde{\theta}$  be the parameter vector of the true distribution of  $(\mathbf{w}, \mathbf{y})$ . It can be shown that, if 1) the sample  $(\mathbf{w}, \mathbf{y})$  is a sample from the assumed family of distributions, 2) the family of distributions under consideration is well behaved, and 3) the sample  $(\mathbf{w}, \mathbf{y})$  is large enough, then the maximum likelihood estimator,  $g_{\text{MLB}}$ , has a Gaussian distribution with mean  $\tilde{\theta}$ , and variance of the form  $1/(nB_{\mathbf{w}, \mathbf{y}}^2)$ , [Wilks 61], where  $n$  (in a sense that is not important for this argument) is the size of the sample, and  $B_{\mathbf{w}, \mathbf{y}}$ , the *Fisher Information*, is a quantity determined solely by  $\tilde{\theta}$  and  $(\mathbf{w}, \mathbf{y})$ .  $g_{\text{MLB}}$  is therefore *consistent*:  $\lim_{n \rightarrow \infty} g_{\text{MLB}}(\mathbf{w}, \mathbf{y}) = \tilde{\theta}$ . Furthermore, it can be shown that no consistent estimator has a lower variance. This means that if the three above conditions hold, no estimator will, on average, provide a closer estimate of the true parameters than will the maximum likelihood estimator. If, in addition, we assume that 4) the performance of a system can not get worse as its parameters get closer to the true parameters, then a system using maximum likelihood estimation will, on average, perform as well as a system using any other form of estimation.

There is also a Bayesian argument for maximum likelihood estimation which is worth considering. From the Bayesian point of view, the true parameter vector is thought of as a random sample from some prior distribution. For the moment, let us assume that the prior is known. In place of the fourth assumption above, assume that 4) the performance of a system can not get worse as its parameter vector becomes more probably the true parameter vector. In this case, clearly, we want to use an estimator,  $g_{\text{MAP}}(\mathbf{w}, \mathbf{y})$ , which has the greatest probability of being the true parameter vector *a posteriori*:

$$\Pr(\Theta = g_{\text{MAP}}(\mathbf{w}, \mathbf{y}) \mid \mathbf{W} = \mathbf{w}, \mathbf{Y} = \mathbf{y}) \triangleq \max_{\theta} \Pr(\Theta = \theta \mid \mathbf{W} = \mathbf{w}, \mathbf{Y} = \mathbf{y}). \quad (2.16)$$

By Bayes' rule, this is equivalent to using a  $g_{\text{MAP}}(\mathbf{w}, \mathbf{y})$  such that

$$\begin{aligned} \Pr_{g_{\text{MAP}}(\mathbf{w}, \mathbf{y})}(\mathbf{W} = \mathbf{w}, \mathbf{Y} = \mathbf{y}) \Pr(\Theta = g_{\text{MAP}}(\mathbf{w}, \mathbf{y})) \\ = \max_{\theta} \Pr_{\theta}(\mathbf{W} = \mathbf{w}, \mathbf{Y} = \mathbf{y}) \Pr(\Theta = \theta). \end{aligned} \quad (2.17)$$

If the prior,  $\Pr(\Theta = \theta)$ , is not known, it is sometimes possible to assume that the prior comes from a given family for which the parameters can be estimated by either using additional training data, or by tying distributions [Jelinek 80], [Brown 83] and [Stern 83]. Another alternative, if there is no information from which to estimate a prior, is simply to assume what is the least informative prior, the prior with the greatest entropy, the uniform distribution. Substituting a uniform prior into (2.17), we find that  $g_{\text{MAP}}(\mathbf{w}, \mathbf{y})$  is such that

$$\Pr_{g_{\text{MAP}}(\mathbf{w}, \mathbf{y})}(\mathbf{W} = \mathbf{w}, \mathbf{Y} = \mathbf{y}) = \max_{\theta} \Pr_{\theta}(\mathbf{W} = \mathbf{w}, \mathbf{Y} = \mathbf{y}). \quad (2.18)$$

Maximum *a posteriori* estimation with a uniform prior, is just maximum likelihood estimation.

In summary, if we assume that: 1) the family of distributions of  $\mathbf{y}$  given  $\mathbf{w}$  is known, 2) the training sample is large, 3) the true language model is known, and 4) the performance of a system will not deteriorate as the parameters become closer to, or are more probably, the true parameters, then choosing an acoustic parameter vector,  $\theta$ , to maximise  $\Pr_{\theta}(\mathbf{Y} = \mathbf{y} \mid \mathbf{W} = \mathbf{w})$  for training data  $(\mathbf{w}, \mathbf{y})$  is an optimal method of estimation for speech recognition.

### 2.3 Maximum Mutual Information Estimation

Let us consider the validity of the premises which justify maximum likelihood estimation.

The first premise states that the true distribution of speech is one of the assumed family of distributions. An accurate model of speech must accurately model the vocal tract, the mouth, the nasal cavity, the lips, and the acoustic channel between the lips and the microphone. At the present time, we are not even close to being able to accurately model such a complex acoustic device. Any family of distributions based solely on true assumptions will have to have an enormous number of free parameters. If, as the second premise requires, the training sample from which these parameters are to be estimated is large enough to apply asymptotic theory to estimation arguments, an unreasonable amount of training data will be required. In other words, our understanding of speech is at such a primitive stage that if we are to use a model whose parameters can be reliably estimated, we will have to include some assumptions about speech which are simply false, thereby invalidating the first premise. The third premise states that the true language model is known. Our knowledge of the syntax and semantics of natural languages, such as English, is at an even more primitive stage than our knowledge of the acoustics of speech. As a result, in a natural-language speech recognition system we will not know the true language model, and this third premise will not be valid. The fourth premise states that the performance of a system will not deteriorate as the parameters become closer to the true parameters. While this assumption is, on the whole, probably true, it is unclear exactly what it means if we are considering a family of distributions in which there are no true parameters.

In the first section of this chapter, we argued that given a language model the goal of acoustic modeling was to develop a model of speech which maximises the mutual information between the sample speech signal  $y$  and the sample word sequence  $w$ . We saw that if the language model is correct, the true acoustic model would maximise this information. This then led to the investigation of a method of parameter estimation, maximum likelihood estimation, which is designed to find parameters which are as close to the true parameters as possible. However, as we can now see, the justification for this method is based on premises which, at the present time, are simply not valid. But there is nothing wrong with our original goal of maximising the mutual information between the sample acoustic signal and the sample word sequence. In this section we consider a method of parameter estimation, maximum mutual information estimation (MMIE) which attempts to choose

parameters that maximize this mutual information, regardless of whether the language model and family of acoustic models are correct.

Suppose that a language model,  $\ell$ , is given. We would like to choose a vector of acoustic parameters,  $\theta$ , to maximize

$$I_{\ell, \theta}(W; Y) = \sum_{w, y} \Pr(W = w, Y = y) \log \left( \frac{\Pr_{\ell, \theta}(W = w, Y = y)}{\Pr_{\ell}(W = w) \Pr_{\ell, \theta}(Y = y)} \right), \quad (2.19)$$

where the  $\ell$  and  $\theta$  subscripts indicate which models and parameters are used in the computation of the subscripted probabilities. Since we do not know  $\Pr(W = w, Y = y)$ , we must instead assume that our sample  $(w, y)$  is representative and choose  $\theta$  to maximize

$$\begin{aligned} f_{\text{MMIE}}(\theta) &= \log \left( \frac{\Pr_{\ell, \theta}(W = w, Y = y)}{\Pr_{\ell}(W = w) \Pr_{\ell, \theta}(Y = y)} \right) \\ &= \log \Pr_{\theta}(Y = y | W = w) - \log \Pr_{\ell, \theta}(Y = y). \end{aligned} \quad (2.20)$$

Choosing  $\theta$  to maximize the first term on the right is the same as finding the maximum likelihood estimate of  $\theta$ . The difference between MLE and MMIE is in the second term.

To understand how this second term makes a difference, let us first expand it in terms of the language model,  $\ell$ , which is assumed given, and the acoustic parameter vector,  $\theta$ ,

$$\Pr_{\ell, \theta}(Y = y) = \sum_{\hat{w}} \Pr_{\theta}(Y = y | W = \hat{w}) \Pr_{\ell}(W = \hat{w}). \quad (2.21)$$

Letting

$$Q_{\theta}(y | w) \triangleq \frac{\partial \Pr_{\theta}(Y = y | W = w)}{\partial \theta_i}, \quad (2.22)$$

consider the derivative of  $f_{\text{MMIE}}(\theta)$  with respect to  $\theta_i$ :

$$\begin{aligned} \frac{\partial f_{\text{MMIE}}(\theta)}{\partial \theta_i} &= \frac{Q_{\theta}(y | w)}{\Pr_{\theta}(Y = y | W = w)} - \frac{\sum_{\hat{w}} Q_{\theta}(y | \hat{w}) \Pr_{\ell}(W = \hat{w})}{\Pr_{\ell, \theta}(Y = y)} = \\ &= Q_{\theta}(y | w) \left[ \frac{1}{\Pr_{\theta}(Y = y | W = w)} - \frac{\Pr_{\ell}(W = w)}{\Pr_{\ell, \theta}(Y = y)} \right] - \sum_{\hat{w} \neq w} \frac{Q_{\theta}(y | \hat{w}) \Pr_{\ell}(W = \hat{w})}{\Pr_{\ell, \theta}(Y = y)}. \end{aligned} \quad (2.23)$$

Compare this expression to the derivative of the objective function used in maximum likelihood estimation,  $f_{\text{MLB}} = \Pr_{\theta}(Y = y | W = w)$ ,

$$\frac{\partial f_{\text{MLB}}}{\partial \theta_i} = Q_{\theta}(y | w). \quad (2.24)$$

The first term in the MMIE derivative is in the same direction as the MLE derivative. The role of the second term is to subtract a component in the direction of  $Q_{\theta}(y | \hat{w})$  for each

incorrect word sequence  $\hat{w} \neq w$ . In MLE, one tries to increase  $\Pr_{\theta}(Y = y | W = w)$  for the correct word sequence  $w$ . In MMIE, one similarly tries to increase  $\Pr_{\theta}(Y = y | W = w)$ , but also tries to decrease  $\Pr_{\theta}(Y = y | W = \hat{w})$  for every incorrect word sequence  $\hat{w} \neq w$ . This indicates a fundamental difference between MLE and MMIE; in MLE only the correct word sequence comes into play, in MMIE every word sequence is taken into account. Furthermore, the greater the prior probability,  $\Pr_l(W = \hat{w})$ , of a word sequence  $\hat{w}$ , the more important it is in determining the MMIE  $\theta$ . This makes good sense, because the greater the probability that the system assigns to a sequence of words,  $\hat{w}$ , a priori, the greater the chance the system will misrecognize  $w$  as  $\hat{w}$ . In MLE, a set of parameters is chosen to maximize the probability of generating the sample acoustic data given the corresponding sample word sequence. In MMIE, a set of parameters are chosen with the explicit aim of discriminating the correct word sequence from every other word sequence.

Arthur Nadas [Nadas 83] points out that if the language model and the distribution family assumed in the acoustic models are correct, then both MLE and MMIE will be consistent estimators, but that MMIE will have a greater variance. This implies that if the language model and the acoustic distribution family are close to correct, and there is limited training data, then MLE will outperform MMIE. If, on the other hand, either is inaccurate, we might suspect that MMIE will outperform MLE, since MMIE does not presuppose model accuracy. Furthermore with a large amount of training data, we would always expect MMIE to perform as well as MLE, since the variances of both estimators vanish as the amount of training data becomes infinite.

We can test these predictions empirically in an experiment suggested by both Arthur Nadas [Nadas 86] and Nick Patterson [Patterson 86]. Consider the very simple problem of a 2-input 2-output decoder. Let the inputs be  $y_1$  and  $y_2$ , and the outputs be  $w_1$  and  $w_2$ . The problem is completely characterised by the language model parameter  $\Pr(W = w_1)$ , and the two acoustic model parameters  $\Pr(Y = y_1 | W = w_1)$  and  $\Pr(Y = y_2 | W = w_2)$ , since  $\Pr(W = w_2) = 1 - \Pr(W = w_1)$ ,  $\Pr(Y = y_2 | W = w_1) = 1 - \Pr(Y = y_1 | W = w_1)$ , and  $\Pr(Y = y_1 | W = w_2) = 1 - \Pr(Y = y_2 | W = w_2)$ . Suppose that these parameters have the following values:

$$\Pr(W = w_1) = .6 \quad \Pr(Y = y_1 | W = w_1) = .6 \quad \Pr(Y = y_2 | W = w_2) = .8. \quad (2.25)$$

The resulting joint probabilities are shown in Table 2.1.

There are only four possible decoders in this problem. From equation (2.8), we can see

	$y_1$	$y_2$
$w_1$	.36	.24
$w_2$	.08	.32

Table 2.1: Joint Probabilities of Words and Acoustic Signals

that the optimal decoder,  $f_{\text{OPT}}$ , will be such that

$$f_{\text{OPT}}(y_1) = w_1 \quad \text{and} \quad f_{\text{OPT}}(y_2) = w_2. \quad (2.26)$$

From equation (2.6), we can see that the performance of  $f_{\text{OPT}}$  is

$$v(f_{\text{OPT}}) = \sum_y \Pr(Y = y \mid W = f_{\text{OPT}}(y)) \Pr(W = f_{\text{OPT}}(y)) = .68. \quad (2.27)$$

The acoustic parameter estimation task is to find estimates for  $\Pr(Y = y_1 \mid W = w_1)$  and  $\Pr(Y = y_2 \mid W = w_2)$  from a sample of  $w$  and  $y$  pairs, assuming some language model parameter  $\Pr_L(W = w_1)$ . We can compare MLE and MMIE on this task by randomly generating training samples, computing MLE and MMIE acoustic parameter estimates, and then using these estimates to compute decoding accuracies. By doing this for a large number of sample sets we can compare the average performance of an MLE decoder and an MMIE decoder.

First, suppose the true language model is known. Table 2.2 shows how MLE and MMIE compare for varying sample sizes. As expected, the performance of both estimators improves with increasing sample size. Furthermore, MLE outperforms MMIE until they both have converged to optimal performance at 1024  $(w, y)$  pairs per sample.

Sample Size	MLE Performance	MMIE Performance
4	.620	.611
8	.637	.632
16	.659	.646
32	.666	.658
64	.674	.663
128	.678	.671
256	.680	.677
512	.680	.679
1024	.680	.680

Table 2.2: MLE vs. MMIE, Assuming True Language Model is Known



Now, suppose that our estimate of the language-model parameter is way off. The joint probabilities computed with a language model in which  $\Pr_{\ell}(W = w_1) = .2$  are displayed in Table 2.3.

	$y_1$	$y_2$
$w_1$	.12	.08
$w_2$	.16	.64

**Table 2.3:** Joint Probabilities Computed with Incorrect Language Model

Performance results using this inaccurate language model are shown in Table 2.4. As in the previous example the performance of MMIE improves with increasing sample size until it reaches the optimal performance of 68 percent accuracy. The performance of MLE, on the other hand, actually decreases with increasing sample size, and is worse than the MMIE performance for all sample sizes greater than 4. Because the decoder is using the wrong language model parameter, as MLE homes in on a better estimate of the true acoustic model parameters, it homes in on the sub-optimal decoder

$$f_{MLE}(y_1) = w_2 \quad \text{and} \quad f_{MLE}(y_2) = w_2. \quad (2.28)$$

By examining Table 2.3 we can see that this sub-optimal decoder has a performance of

$$v(f_{MLE}) = \sum_y \Pr(Y = y \mid W = f_{MLE}(y)) \Pr(W = f_{OPT}(y)) = .40. \quad (2.29)$$

Sample Size	MLE Performance	MMIE Performance
4	.575	.574
8	.539	.609
16	.512	.658
32	.499	.678
64	.474	.680
128	.452	.680
256	.429	.680
512	.409	.680
1024	.402	.680

**Table 2.4:** MLE vs. MMIE, Assuming Incorrect Language Model

The choice, then, of whether to use MLE or MMIE depends on the accuracy of model assumptions and on the amount of training data available relative to the number of parameters to be estimated.

In the chapters which follow, we will explore a particular class of acoustic models, and compare the performance of MLE and MMIE in real speech recognition experiments.

### 3. Hidden Markov Models

#### 3.1 Basics

This section consists of a description of what has become the predominant class of acoustic models, *hidden Markov models*.

A sequence of random variables  $X = X_1, X_2, \dots$  is a *first-order n-state Markov chain* provided that for each  $t$ ,  $X_t$  ranges over the integers 1 to  $n$  and further that

$$\Pr(X_{t+1} = z_{t+1} \mid X_1^t = z_1^t) = \Pr(X_{t+1} = z_{t+1} \mid X_t = z_t), \quad (3.1)$$

where the abbreviation  $X_i^j$  is used to represent the sequence  $X_i, X_{i+1}, \dots, X_j$ . Equation (3.1) is known as the *Markov assumption*. In words, it states that the probability that the Markov chain will be in a particular state at a given time depends only on the state of the Markov chain at the previous time.

The parameters of a Markov chain are

$$a_{ij} \triangleq \Pr(X_{t+1} = j \mid X_t = i), \quad (3.2)$$

and

$$c_i \triangleq \Pr(X_1 = i). \quad (3.3)$$

We say that a transition from state  $i$  to state  $j$  occurred at time  $t$  if the Markov chain was in state  $i$  at time  $t$  and in state  $j$  at time  $t + 1$ . The *transition probability*  $a_{ij}$  is the probability of making the transition to state  $j$  at time  $t$  given that the chain was in state  $i$  at time  $t$ . In a *stationary* Markov chain, we assume that this probability is independent of the time at which the transition occurs. The *initial-state probability*  $c_i$  is the probability that the Markov chain will start in state  $i$ . Since the initial-state probabilities and the transition probabilities are probabilities, they must all be non-negative, and for every state,  $i$ , they must satisfy the following constraints:

$$\sum_{j=1}^n a_{ij} = 1, \quad \text{and} \quad \sum_{j=1}^n c_j = 1. \quad (3.4)$$

In a *hidden Markov model* a sequence of random variables  $Y$  is a probabilistic function of a stationary Markov chain  $X$ . There are two cases to consider.  $X$  is a hidden Markov

model for a sequence of discrete random variables,  $Y$ , provided that for each  $t$ ,  $Y_t$  ranges over a discrete space, and further that

$$\Pr(Y_t = y_t \mid Y_t^{t-1} = y_t^{t-1}, X_t^T = x_1^T) = \Pr(Y_t = y_t \mid X_t = x_t, X_{t+1} = x_{t+1}). \quad (3.5)$$

$X$  is a hidden Markov model for a sequence of continuous random variables  $Y$ , provided that for each  $t$ ,  $Y_t$  ranges over a continuous space, such as  $\mathbb{R}^N$ , and further that

$$f(Y_t = y_t \mid Y_t^{t-1} = y_t^{t-1}, X_t^T = x_1^T) = f(Y_t = y_t \mid X_t = x_t, X_{t+1} = x_{t+1}), \quad (3.6)$$

where  $f(Y_t = y_t)$  is the probability density at  $y_t$ .

The sequence  $y_1, y_2, \dots$  is the observed output of the hidden Markov model. The state sequence  $x_1^T$  is not observed; it is hidden. In this thesis, equation (3.5) or its equivalent (3.6) will be referred to as the *output-independence assumption*. It states that the probability that a particular output will occur at a given time depends only on the transition taken at that time.

In addition to the initial-state probabilities and the transition probabilities, the parameters of a hidden Markov model for a discrete sequence include the output probabilities

$$b_{ij}(k) \triangleq \Pr(Y_t = k \mid X_t = i, X_{t+1} = j). \quad (3.7)$$

Each  $b_{ij}(k)$  must be non-negative, and for each transition  $i \rightarrow j$ ,

$$\sum_k b_{ij}(k) = 1. \quad (3.8)$$

In the continuous case, the output distributions in (3.6) are usually parameterized. For example, they may be assumed to be Gaussian. In which case, each output distribution would be parameterized by a mean and covariance matrix. The constraints on the parameters depend on the particular type of output distributions. For Gaussians, for example, the covariance matrices must all be positive-definite. The parameters of a hidden Markov model of a continuous sequence are then the initial-state probabilities, the transition probabilities, and the parameters of the output distributions.

In order to avoid making every point twice, once for the discrete case, and once for the continuous case, let us generalize the  $\Pr$  notation to stand for both the probability of an event, and for the probability density at a point:

$$\Pr(Y_t = y_t) \triangleq \begin{cases} \Pr(Y_t = y_t), & \text{in the discrete case;} \\ f(Y_t = y_t), & \text{in the continuous case.} \end{cases} \quad (3.9)$$

In the same way let us also use the term  $b_{ij}(y_t)$  to refer to both a probability in the discrete case and a probability density in the continuous case:

$$b_{ij}(y_t) \triangleq \begin{cases} \Pr(Y_t = y_t | X_t = i, X_{t+1} = j), & \text{in the discrete case;} \\ f(Y_t = y_t | X_t = i, X_{t+1} = j) & \text{in the continuous case.} \end{cases} \quad (3.10)$$

I shall refer to the *probability of generating* some event or sequence of events,  $y$ , in the discrete case to mean the probability of sampling  $y$ , and in the continuous case to mean the probability density at  $y$ . Observations which are clearly vectors of continuous parameters will be denoted by bold-faced variables. Discrete observations will be denoted by ordinary variables. Generic observations, which may be either from a discrete or a continuous space will also be denoted by ordinary variables. Variables which denote whole sequences will always be in a bold-faced type, except when the bounds are listed explicitly as in  $y_1^T$ .

The Markov assumption and the output-independence assumption are made so that the number of parameters in a hidden Markov model will be relatively small, and so that certain computations can be performed rapidly. An important example is the computation of the probability that a hidden Markov model will generate a particular sequence  $y_1^T$ . Since we do not know which path was taken through the model when generating  $y_1^T$ , we must sum over all possible paths containing  $T$  transitions:

$$\Pr(Y_1^T = y_1^T) = \sum_{z_1^{T+1}} \Pr(X_1^{T+1} = z_1^{T+1}) \Pr(Y_1^T = y_1^T | X_1^{T+1} = z_1^{T+1}). \quad (3.11)$$

The computation of  $\Pr(X_1^T = z_1^T)$  can be simplified with the Markov assumption:

$$\begin{aligned} \Pr(X_1^T = z_1^T) &= \Pr(X_1 = z_1) \prod_{t=1}^{T-1} \Pr(X_{t+1} = z_{t+1} | X_t = z_t) \\ &= \Pr(X_1 = z_1) \prod_{t=1}^{T-1} \Pr(X_{t+1} = z_{t+1} | X_t = z_t). \end{aligned} \quad (3.12)$$

The computation of  $\Pr(Y_1^T = y_1^T | X_1^{T+1} = z_1^{T+1})$  can be simplified with the output-independence assumption:

$$\begin{aligned} &\Pr(Y_1^T = y_1^T | X_1^{T+1} = z_1^{T+1}) \\ &= \Pr(Y_1 = y_1 | X_1^{T+1} = z_1^{T+1}) \prod_{t=2}^T \Pr(Y_t = y_t | X_1^{T+1} = z_1^{T+1}, Y_1^{t-1} = y_1^{t-1}) \\ &= \prod_{t=1}^T \Pr(Y_t = y_t | X_t = z_t, X_{t+1} = z_{t+1}). \end{aligned} \quad (3.13)$$

Substituting (3.12) and (3.13) into (3.11), we find that  $\Pr(Y_1^T = y_1^T)$  is equal to

$$\sum_{x_1^{T+1}} \Pr(X_1 = x_1) \prod_{t=1}^T \Pr(X_{t+1} = x_{t+1} | X_t = x_t) \Pr(Y_t = y_t | X_t = x_t, X_{t+1} = x_{t+1}). \quad (3.14)$$

Since the number of paths of length  $T$  through a Markov model grows exponentially with  $T$ , it may appear that the computation in (3.11) grows exponentially with  $T$ . However, because each factor  $\Pr(X_{t+1} = x_{t+1} | X_t = x_t) \Pr(Y_t = y_t | X_t = x_t, X_{t+1} = x_{t+1})$  only involves  $y_t, x_t$ , and  $x_{t+1}$ ,  $\Pr(Y_1^T = y_1^T)$  can be computed recursively in linear time. Let

$$\alpha_i(t) \triangleq \Pr(Y_1^t = y_1^t, X_{t+1} = i). \quad (3.15)$$

Then for all states,  $i$ ,

$$\alpha_i(0) = \Pr(X_1 = i) = c_i, \quad (3.16)$$

and for  $t > 0$

$$\alpha_i(t) = \sum_j \alpha_j(t-1) a_{ji} b_{ji}(y_t). \quad (3.17)$$

Clearly  $\Pr(Y_1^T = y_1^T) = \sum_i \alpha_i(T)$ .

Another important example of a problem that can be solved quickly by taking advantage of the Markov and output-independence assumptions is the determination of the most probable state sequence given a particular output sequence  $y_1^T$ . Similar to the definition of  $\alpha_i(t)$  given above, define  $\nu_i(t)$  to be the joint probability of taking the most probable state sequence ending at state  $i$  at time  $t+1$ , and generating  $y_1^t$ .  $\nu$  can be computed in linear time with the following recursion:

$$\nu_i(0) = c_i \quad (3.18)$$

$$\nu_i(t) = \max_j \nu_j(t-1) a_{ji} b_{ji}(y_t). \quad (3.19)$$

$\rho_i(t)$ , the most probable state sequence ending in state  $i$  at time  $t+1$  given  $y_1^t$  can then be computed in linear time by noting that

$$\rho_i(t) = \rho_{\pi_i(t)}(t-1) \quad \parallel \quad i, \quad (3.20)$$

where

$$\pi_i(t) \triangleq \operatorname{argmax}_j \nu_j(t-1) a_{ji} b_{ji}(y_t), \quad (3.21)$$

and  $||$  denotes sequence concatenation. The most probable state sequence given  $y_1^T$  is then  $\rho_k(T)$ , where

$$k \triangleq \underset{i}{\operatorname{argmax}} \nu_i(T). \quad (3.22)$$

This algorithm was discovered by Bellman [Bellman 57]. It was then rediscovered and applied to problems in information theory by Viterbi [Viterbi 67]. I shall refer to the resultant alignment between output sequence and transition sequence as a *Viterbi alignment*.

### 3.2 Modifications to the Basic Model

In this section we shall consider a few simple modifications to the basic hidden Markov model that is described in the previous section. These modifications do not extend the class of probability distributions beyond the basic model, but are introduced because they are convenient when applying hidden Markov models to speech recognition.

The number of states in a Markov chain is a measure, albeit crude, of the complexity of the finite-state grammar represented by that chain. As Jim Baker points out, the modeling of speech by a hidden Markov model should not be regarded as a statement that the intricacies of speech are best described by a finite-state grammar, but rather the intricacies of speech should be regarded as a "prescription to be followed in the formulation of the state space", [Baker 79].

In succeeding sections, we will discuss the estimation of hidden Markov model parameters from sample speech. As in all kinds of parameter estimation, the number of parameters that can be reliably estimated is a function of the size of the training sample. Therefore, with a limited training sample, it is necessary to limit the number of parameters in the model.

On the one hand, then, we would like to use a large number of states and transitions in an underlying Markov chain in order to model complex phonetic events. But on the other hand, we would like to keep the number of transition probabilities and output distributions small because we are faced with limited training data. One way to address these conflicting requirements is to impose constraints which force transition probability distributions on sets of transitions originating at different states to be the same, and which force output distributions on different transitions to be the same. By doing this, the state space and number of transitions can be made arbitrarily large, while the number of parameters in the model can be kept arbitrarily small.

Two transition probabilities are *tied* if they are constrained to be equal to one another. Denote the set of all transitions which have transition probabilities that are tied to the transition probability on the transition  $i \rightarrow j$  by  $\mathcal{T}_{ij}$ . The distribution of transition probabilities at state  $i$  is tied to the distribution of transition probabilities at state  $j$  if there is a permutation,  $\pi$ , such that for all  $k$ ,  $a_{ik}$  is tied to  $a_{j\pi(k)}$ . Denote the set of all states which have transition probability distributions that are tied to the transition probability distribution at state  $i$  by  $\mathcal{S}_i$ . The output probability distributions on transitions  $i \rightarrow j$  and  $k \rightarrow l$  are tied if for all outputs,  $y_t$ ,  $\Pr(Y_t = y_t \mid X_t = i, X_{t+1} = j)$  is constrained to be equal to  $\Pr(Y_t = y_t \mid X_t = k, X_{t+1} = l)$ . Denote the set of all transitions which have output probability distributions that are tied to the output probability distribution on the transition  $i \rightarrow j$  by  $\mathcal{D}_{ij}$ . Tying induces an equivalence relation on transition probabilities, on transition probability distributions, and on output probability distributions in the obvious ways.

Another way of reducing the number of free parameters in a hidden Markov model is simply to require that certain parameters have specified values. We may, for example, require that certain transition probabilities be zero, thereby prohibiting the associated transitions. Similarly, by fixing certain initial-state probabilities at zero, we can ensure that all state sequences with non-zero probability begin in some set of *initial states*.

We will also want to insist that all state sequences end in a set of final states. The set of final states in a Markov model are all those states from which all transitions are prohibited. The probability of a state sequence  $\mathbf{z}_1^T$  is zero unless  $\mathbf{z}_t$  is a final state. A Markov model with at least one reachable final state then defines a probability distribution on state sequences of varying lengths.

In models with prohibited transitions, we can often further reduce the number of allowed transitions required to adequately model an acoustic process by using *null transitions*, which allow the model to change state without producing any output. The incorporation of null transitions into hidden Markov models is discussed in detail in [Bahl 83]. If certain transition probabilities are required to be zero, then, when computing the probability of a sequence, we need not sum over states sequences that involve those transitions. When this savings is taken into account, it can be seen that the computation of the probability of a sequence is linear in the number of transitions, null or ordinary, with non-zero transition probabilities.

Throughout this thesis, diagrams of hidden Markov models will be presented. In these figures, transitions that have their transition probabilities fixed at 0 will not be shown, and null transitions will be represented by dashed arrows. There will usually be one initial state,



which will be the state at the far left, and one final state, which will be the state at the far right. For example, in figure 3.1 the transitions  $1 \rightarrow 1$  and  $2 \rightarrow 2$  are ordinary transitions. The transitions  $1 \rightarrow 3$ ,  $2 \rightarrow 1$ ,  $3 \rightarrow 1$ ,  $3 \rightarrow 2$ , and  $3 \rightarrow 3$  have transition probabilities that are fixed at 0. State 1 is the single initial state, and state 3 is the single final state.

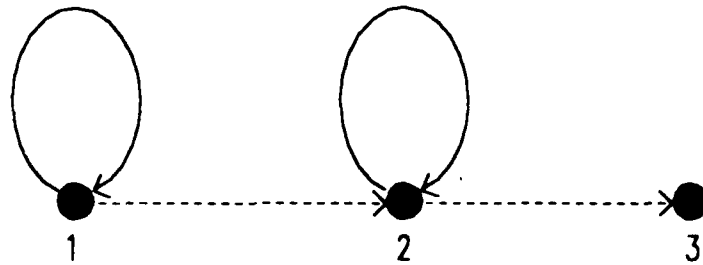


Figure 3.1: Sample Hidden Markov Model Diagram

Thus far, we have referred to a transition in a Markov model as a pair of states, an origin and a destination, which makes sense because in an ordinary Markov chain it is of no use to contemplate more than one transition between two states. On the other hand, in a hidden Markov model with parametric distributions it does make sense to consider two separate transitions between a pair of states. Consider a hidden Markov model in which all output distributions are assumed to be Gaussian, for example. If there are two different transitions with different output distributions between the states  $i$  and  $j$  then the output probability distribution given that a transition from  $i$  to  $j$  has occurred is a mixture of two Gaussians. We can thus implement output distributions which are mixtures in a straightforward manner by allowing multiple transitions between states.

While these modifications to the basic hidden Markov model are important in the application of hidden Markov models to speech recognition, with the exception of tying, they are not crucial to the issues which will be addressed in the rest of this thesis. Furthermore, although it is straightforward to incorporate fixed probabilities, null transitions and multiple transitions between states into the models we will be discussing, it significantly complicates many of the formulas relating to these models. As a result, we will only amend our basic model by the inclusion of tied probabilities and tied distributions. It is left as an exercise for the interested reader to include the other modifications.

When applying hidden Markov models to speech recognition, it will be convenient to consider a family of models. There might be one model for each word, for example. Let  $M = (m_1, m_2, \dots, m_r)$  be a family of hidden Markov models. The parameter vector of  $M$

is defined to be  $\Theta = (\theta_1, \theta_2, \dots, \theta_r)$  where  $\theta_i$  is the parameter vector of  $m_i$ . The concept of tying is extended to include probabilities and probability distributions from different members of the family. Thus, we say that the transition probability on transition  $i \rightarrow j$  in model  $m$  is tied to the transition probability  $k \rightarrow l$  in model  $m'$ , if  $a_{ij}$  is constrained to be equal to  $a'_{kl}$ . We extend the concept of tied transition probability distributions and tied output distributions in the same manner. In addition, we say that the initial-state probability of state  $i$  in model  $m$  is tied to the initial-state probability of state  $j$  in model  $m'$  if  $c_i$  is constrained to be equal to  $c'_j$ . Because of tying or other constraints on  $\Theta$ , it may be that any change to the parameter vector of one of the members of a family, say  $m$ , will necessitate a change to the parameter vector of some other member of the family, say  $m'$ . When this is the case, we say that  $m$  entails  $m'$ . A member,  $m$ , is *representative* of a family if it is entailed by each member of the family. A family which has a representative member is said to be *close-knit* [Bahl 86].

The concept of a close-knit family is important when estimating the parameter vector of a family of hidden Markov models. This is because the parameters for all members of a close-knit family can be estimated from a sample which is assumed to have been generated by a representative member. If we have a method of estimating the parameters for that representative member, then we have a way of estimating parameters for the whole family.

### 3.3 The Forward-Backward Algorithm

Hidden Markov models owe their current popularity to the existence of a fast algorithm for computing maximum likelihood estimates of their parameters, the *forward-backward algorithm*. In this section, we will describe this algorithm.

Let us first consider the problem of finding the maximum likelihood estimates of the transition probabilities in an ordinary Markov chain. In this case, we simply observe in the training sample the state sequence and hence the transition sequence. The probability of generating the training data is just the product of the transition probabilities for the transitions reflected in the sample. Let  $n_{ij}$  be the number of times a transition with a transition probability that is tied to the transition probability on the transition from state  $i$  to state  $j$  occurs in the training data. It is easy to show that the maximum likelihood

estimate for the transition probability  $a_{ij}$  is

$$\hat{a}_{ij} = \frac{n_{ij}}{\sum_k n_{ik}}. \quad (3.23)$$

What makes MLE for hidden Markov models more complicated is that in a hidden Markov model, the underlying state sequence is not observed, and it is therefore not possible to count how many times each transition occurred. If we knew the sequence of transitions, then we could estimate the transition probabilities for a hidden Markov model as they are estimated for an ordinary Markov chain, and we could estimate the parameters in an output distribution,  $d$ , from the collection of all  $y_i$  which were generated while a transition with an output distribution tied to  $d$  occurred. The problem is we don't know what we need to know to obtain the maximum likelihood parameter estimates.

This problem is an instance of the general problem of finding maximum likelihood estimates when the data observed are incomplete. Suppose we observe training data,  $y$ . In order to determine the vector of parameters,  $\hat{\theta}$ , which maximizes  $\Pr_{\hat{\theta}}(Y = y)$ , we would also like to know some additional data,  $z$ . The method we will use will be to assume a vector of parameters  $\theta$  and estimate the probability that each  $z$  occurred in the generation of  $y$ . We can then pretend that we had in fact observed an  $(z, y)$  pair with frequency proportional to the probability that  $z$  occurred, given  $y$  and our assumed parameter vector,  $\theta$ , to compute a new vector of parameter estimates,  $\hat{\theta}$ . We can then let  $\theta$  be this new vector of estimates and repeat the process, in an effort to iteratively improve our estimates. In a very important theorem, Leonard Baum proved that  $\Pr_{\hat{\theta}}(Y = y) \geq \Pr_{\theta}(Y = y)$ , with equality if and only if  $\hat{\theta} = \theta$  [Baum 72]. This technique has since been adopted by other statisticians and is referred to as the EM algorithm [Dempster 77]. We will now prove this theorem using a version of Baum's proof which has been presented by Issac Meilijson [Meilijson 85].

First we introduce the unobserved data,  $z$ , into our objective function:

$$\begin{aligned} \Pr_{\hat{\theta}}(Y = y) &= \Pr_{\hat{\theta}}(Y = y) \left( \frac{\Pr_{\hat{\theta}}(X = z, Y = y)}{\Pr_{\hat{\theta}}(X = z, Y = y)} \right) \\ &= \frac{\Pr_{\hat{\theta}}(X = z, Y = y)}{\Pr_{\hat{\theta}}(X = z | Y = y)}. \end{aligned} \quad (3.24)$$

Now take the conditional expectation of  $\log \Pr_{\hat{\theta}}(Y = y)$  over  $X$  using an assumed vector of parameters  $\theta$ :

$$\log \Pr_{\hat{\theta}}(Y = y) = \log \Pr_{\hat{\theta}}(X = z, Y = y) - \log \Pr_{\hat{\theta}}(X = z | Y = y), \quad (3.25)$$

$$\begin{aligned} E_{\theta} [\log \Pr_{\hat{\theta}}(Y = y)]_{X|Y=y} &= \sum_z \Pr_{\theta}(X = z | Y = y) \log \Pr_{\hat{\theta}}(Y = y) \\ &= \log \Pr_{\hat{\theta}}(Y = y), \end{aligned} \quad (3.26)$$

where  $E_{\theta} [f]_{X|Y=y}$  denotes the expectation of the function  $f$  over the random variable  $X$ , given that  $Y = y$ , and computed with the parameter vector  $\theta$ . Therefore,

$$\log \Pr_{\hat{\theta}}(Y = y) = E_{\theta} [\log \Pr_{\hat{\theta}}(X, Y = y)]_{X|Y=y} - E_{\theta} [\log \Pr_{\hat{\theta}}(X | Y = y)]_{X|Y=y}. \quad (3.27)$$

By a derivation similar to (2.5) we find that

$$\begin{aligned} E_{\theta} [\log \Pr_{\hat{\theta}}(X | Y = y)]_{X|Y=y} &= \sum_x \Pr_{\theta}(X = x | Y = y) \log \Pr_{\hat{\theta}}(X = x | Y = y) \\ &\leq \sum_x \Pr_{\theta}(X = x | Y = y) \log \Pr_{\theta}(X = x | Y = y) \\ &= E_{\theta} [\log \Pr_{\theta}(X | Y = y)]_{X|Y=y}, \end{aligned} \quad (3.28)$$

with equality if and only if  $\theta = \hat{\theta}$ . Therefore, if we choose  $\hat{\theta}$  so that

$$E_{\theta} [\log \Pr_{\hat{\theta}}(X, Y = y)]_{X|Y=y} \geq E_{\theta} [\log \Pr_{\theta}(X, Y = y)]_{X|Y=y}, \quad (3.29)$$

then

$$\log \Pr_{\hat{\theta}}(Y = y) \geq \log \Pr_{\theta}(Y = y). \quad (3.30)$$

This proves that the following algorithm will converge to a local maximum, or at least to a saddle point, of the likelihood function.

1. Guess an initial vector of parameters,  $\theta$ .
2. Choose  $\hat{\theta}$  to maximise  $E_{\theta} [\log \Pr_{\hat{\theta}}(X, Y = y)]_{X|Y=y}$ .
3. Set  $\theta$  to be  $\hat{\theta}$ .
4. If a convergence criteria is not met, go back to step 2.

Let us now consider the application of the EM algorithm to hidden Markov models. The observed training sample  $y$  is simply the output of the model. The unobserved data  $x$  is the state sequence, or transition sequence, taken while generating  $y$ . In step 1, we assume a vector of parameters,  $\theta$ , consisting of initial-state probabilities, transition probabilities, and output distribution parameters. Step 2, can be performed in two parts. First, at every time, compute the probability that each transition is taken at that time while generating  $y$ . Second, use these probabilities to find a new vector of parameters,  $\hat{\theta}$ , which maximises  $E_{\theta} [\log \Pr_{\hat{\theta}}(X, Y = y)]_{X|Y=y}$ .

As in the previous section, let

$$\alpha_i(t) \triangleq \Pr(Y_1^t = y_1^t, X_{t+1} = i). \quad (3.31)$$

Let

$$\beta_i(t) \triangleq \Pr(Y_{t+1}^T = y_{t+1}^T | X_{t+1} = i). \quad (3.32)$$

Then

$$\Pr(Y_1^T = y_1^T, X_t = i, X_{t+1} = j) = \alpha_i(t-1) a_{ij} b_{ij}(y_t) \beta_j(t). \quad (3.33)$$

In words, the probability that a particular transition  $i \rightarrow j$  is taken at time  $t$ , and that the entire training sequence  $y_1^T$  is generated is equal to the probability of generating the sequence  $y_1^{t-1}$  and arriving at state  $i$ , times the probability of taking the transition  $i \rightarrow j$ , times the probability of generating  $y_t$  while taking the transition  $i \rightarrow j$ , times the probability of generating the sequence  $y_{t+1}^T$  having started at state  $j$ . We have seen, in section 3.1, that all the  $\alpha_i(t)$ 's can be computed in linear time. There is also a recursion which allows the  $\beta_i(t)$ 's to be computed in linear time: for all states,  $i$ ,

$$\beta_i(T) = 1, \quad (3.34)$$

and for  $t, 0 \leq t < T$ ,

$$\beta_i(t) = \sum_j a_{ij} b_{ij}(y_{t+1}) \beta_j(t+1). \quad (3.35)$$

In the first part of the second step above, we need to compute the probability that a transition  $i \rightarrow j$  occurred at time  $t$  given that the model generates  $y_1^T$ . Denote this probability by

$$\begin{aligned} \gamma_{ij}(t) &\triangleq \Pr(X_t = i, X_{t+1} = j \mid Y_1^T = y_1^T) \\ &= \frac{\alpha_i(t-1) a_{ij} b_{ij}(y_t) \beta_j(t)}{\sum_k \alpha_k(T)}. \end{aligned} \quad (3.36)$$

Using the above recursions for the  $\alpha$ 's and  $\beta$ 's, these probabilities can be computed in time which is linear in  $T$  [Baum 72].

We now want to use these probabilities which have been computed using a vector of parameters,  $\theta$ , to choose a vector  $\hat{\theta}$  which maximizes

$$E_{\theta} [\log \Pr_{\hat{\theta}}(X, Y = y)]_{X|Y=y} = \sum_{\mathbf{x}} \Pr_{\theta}(X = \mathbf{x} \mid Y = y) \log \Pr_{\hat{\theta}}(X = \mathbf{x}, Y = y). \quad (3.37)$$

This is equivalent to choosing  $\hat{\theta}$  to maximize

$$\prod_{\mathbf{x}} \Pr_{\hat{\theta}}(X = \mathbf{x}, Y = y)^{\Pr_{\theta}(X = \mathbf{x} \mid Y = y)}. \quad (3.38)$$

It is useful to think of (3.38) as the probability of generating a sequence,  $S_y$ , of independent samples  $(\mathbf{x}, y)$ , each of which occurs  $\Pr_{\theta}(X = \mathbf{x} \mid Y = y)$  times. In other words, there are  $\Pr_{\theta}(X = \mathbf{x} \mid Y = y)$   $(\mathbf{x}, y)$ 's for each state sequence  $\mathbf{x}$  in  $S_y$ . Of course,  $\Pr_{\theta}(X = \mathbf{x} \mid Y = y)$  is not a whole number, so one must think of a sequence occurring a

fractional number of times. The probability of generating  $S_y$  is just the product of a lot of initial state probabilities, transition probabilities, and output probabilities. The number of times a transition,  $i \rightarrow j$ , occurs at time  $t$  in  $S_y$  is just  $\gamma_{ij}(t)$ . We can choose the initial state probabilities to maximize (3.38) by setting each  $\hat{c}_i$  to be proportional to the number of times  $i$  is the first state in  $(x, y)$  samples in  $S_y$ :

$$\hat{c}_i = \frac{\sum_j \gamma_{ij}(1)}{\sum_{i,j} \gamma_{ij}(1)}. \quad (3.39)$$

We can choose the transition probabilities to maximize (3.38) by setting each  $\hat{a}_{ij}$  to be proportional to the number of times a transition in  $T_{ij}$  occurs in  $S_y$ :

$$\hat{a}_{ij} = \frac{\sum_{k \rightarrow l \in T_{ij}} \sum_{t=1}^T \gamma_{kl}(t)}{\sum_{k \in S_i} \sum_l \sum_{t=1}^T \gamma_{kl}(t)}. \quad (3.40)$$

We can maximize (3.38) with respect to  $\phi$ , the vector of parameters of the output distribution associated with transitions that are tied to  $i \rightarrow j$ , by choosing  $\phi$  to maximize the probability, or equivalently the log of the probability, of generating all the  $y_t$  which occur with a transition with an output distribution tied to that on  $i \rightarrow j$  in  $S_y$ . This amounts to choosing  $\phi$  to maximize

$$g(\phi) = \sum_{k \rightarrow l \in \mathcal{D}_{ij}} \sum_{t=1}^T \gamma_{kl}(t) \log \Pr_{\phi}(Y_t = y_t), \quad (3.41)$$

subject to whatever constraints may apply to  $\phi$ .

In the discrete case,  $\phi$  consists of a vector of discrete probabilities. For each discrete output symbol,  $m$ , there is a parameter

$$b_{\phi}(m) = \Pr_{\phi}(Y_t = m). \quad (3.42)$$

The vector of the probabilities must satisfy

$$\sum_m b_{\phi}(m) = 1. \quad (3.43)$$

Using the LaGrange multiplier  $\lambda$  and solving the set of simultaneous equations, one for each  $m$ , of the form

$$\frac{\partial}{\partial b_{\phi}(m)} \left[ \sum_{k \rightarrow l \in \mathcal{D}_{ij}} \sum_{t=1}^T \gamma_{kl}(t) \log \Pr_{\phi}(Y_t = y_t) - \lambda \left( \sum_m b_{\phi}(m) - 1 \right) \right] = 0, \quad (3.44)$$

we find that

$$b_{\phi}(m) = \frac{\sum_{k \rightarrow l \in \mathcal{D}_{ij}} \sum_{t: y_t = m} \gamma_{kl}(t)}{\sum_{k \rightarrow l \in \mathcal{D}_{ij}} \sum_{t=1}^T \gamma_{kl}(t)}. \quad (3.45)$$

Reestimation formulas for the continuous densities that will be discussed in this thesis are presented in the appendices.

### 3.4 Maximum Mutual Information Estimation

Let  $M$  be a close-knit family of hidden Markov models, parameterized by  $\Theta$ . Let  $M$  also denote a random variable ranging over the members of the family  $M$ . In order to keep the formulas short, let us leave the random variables out of the formulas when it is clear what they should be. Thus, we write  $\Pr(m)$  as an abbreviation for  $\Pr(M = m)$ . Let  $\Pr_m(y_1^T)$  denote  $\Pr_{\Theta}(Y_1^T = y_1^T | M = m)$ , the probability of generating  $y_1^T$  from model  $m$  using the parameter vector  $\Theta$ . Suppose that we are given the training sample  $(m, y_1^T)$ , and that  $m$  is representative of the family  $M$ . In maximum mutual information estimation, we would like to choose  $\Theta$  so as to maximize the mutual information between  $m$  and  $y_1^T$ ,

$$\begin{aligned} I_{\Theta}(m, y_1^T) &= \log \frac{\Pr_{\Theta}(Y_1^T = y_1^T, M = m)}{\Pr_{\Theta}(Y_1^T = y_1^T) \Pr(m)} \\ &= \log(\Pr_m(y_1^T)) - \log \sum_{m'} \Pr_{m'}(y_1^T) \Pr(m'). \end{aligned} \quad (3.46)$$

The forward-backward algorithm which was described in the previous section is a hill-climbing algorithm for maximum likelihood estimation. Its primary advantage over gradient descent is that it produces a direction and step size which are guaranteed to improve, or at least not to worsen, the likelihood function. Unfortunately, no such method is known for maximum mutual information estimation, and we must therefore resort to the use of traditional maximization techniques, such as gradient descent.

Let us begin, then, by examining the derivative of  $I_{\Theta}(m, y_1^T)$  with respect to  $\theta$ , a component of  $\Theta$ ,

$$\frac{\partial}{\partial \theta} I_{\Theta}(m, y_1^T) = \frac{\frac{\partial}{\partial \theta} \Pr_m(y_1^T)}{\Pr_m(y_1^T)} - \frac{\sum_{m'} \Pr(m') \frac{\partial}{\partial \theta} \Pr_{m'}(y_1^T)}{\sum_{m'} \Pr(m') \Pr_{m'}(y_1^T)}. \quad (3.47)$$

To compute this derivative, we need to be able to compute the derivative of the probability that a model,  $m$ , will generate  $y_1^T$ ,

$$\frac{\partial \Pr_m(y_1^T)}{\partial \theta} = \frac{\partial}{\partial \theta} \sum_{z_1^{T+1}} \Pr(z_1) \prod_{t=1}^T \Pr(y_t | z_t, z_{t+1}) \Pr(z_{t+1} | z_t). \quad (3.48)$$

Let us suppose, first, that  $\theta$  is a transition probability. Let  $\mathcal{T}_{\theta}$  be the set of all transitions with transition probabilities tied to  $\theta$ . To shorten the formulas which follow, define  $\eta$  as

$$\eta(t1, t2) \triangleq \prod_{i=t1}^{t2} \Pr(y_i | z_i, z_{i+1}) \Pr(z_{i+1} | z_i). \quad (3.49)$$

Letting  $\delta(b)$  be 1 if the Boolean condition  $b$  is true and 0 otherwise,

$$\begin{aligned} \frac{\partial \text{Pr}_m(y_1^T)}{\partial \theta} &= \sum_{x_1^{T+1}} \text{Pr}(x_1) \sum_{t=1}^T \delta(x_t \rightarrow x_{t+1} \in \mathcal{I}_\theta) \eta(1, t-1) \text{Pr}(y_t | x_t, x_{t+1}) \eta(t+1, T) \\ &= \sum_{t=1}^T \sum_{x_1^{T+1}} \text{Pr}(x_1) \delta(x_t \rightarrow x_{t+1} \in \mathcal{I}_\theta) \eta(1, t-1) \text{Pr}(y_t | x_t, x_{t+1}) \eta(t+1, T). \end{aligned} \quad (3.50)$$

Since  $\delta(x_t \rightarrow x_{t+1} \in \mathcal{I}_\theta)$  is non-zero if and only if the transition probability on the  $t$ th transition in  $x_1^{T+1}$  is tied to  $\theta$ , for each time,  $t$ , we are summing over all paths,  $x_1^{T+1}$ , in which the transition probability on the  $t$ th transition is tied to  $\theta$ . We can therefore rearrange this equation and sum over all transitions with a transition probability tied to  $\theta$ , and over all paths which take such a transition at time  $t$ ,

$$\frac{\partial \text{Pr}_m(y_1^T)}{\partial \theta} = \sum_{t=1}^T \sum_{i \rightarrow j \in \mathcal{I}_\theta} \sum_{x_1^{T+1}: x_t=i, x_{t+1}=j} \text{Pr}(x_1) \eta(1, t-1) b_{ij}(y_t) \eta(t+1, T). \quad (3.51)$$

The inside sum is the sum over all state sequences which take the transition  $i \rightarrow j$  at time  $t$  of the joint probability of the state sequence and the output sequence. As we have seen in the last section, this is just the probability of taking the state sequence from an initial state to  $i$  and generating  $y_1^{t-1}$ , times the probability of generating  $y_t$  given the transition  $i \rightarrow j$ , times the probability of taking any sequence from state  $j$  and generating  $y_{t+1}^k$ . Recalling the definitions of  $\alpha$  and  $\beta$ , we have

$$\frac{\partial \text{Pr}_m(y_1^T)}{\partial \theta} = \sum_{t=1}^T \sum_{i \rightarrow j \in \mathcal{I}_\theta} \alpha_i(t-1) b_{ij}(y_t) \beta_j(t). \quad (3.52)$$

Similarly if  $\theta$  is an initial state probability, and  $\mathcal{I}_\theta$  is the set of all initial state probabilities tied to  $\theta$ , then

$$\frac{\partial \text{Pr}_m(y_1^T)}{\partial \theta} = \sum_{i \in \mathcal{I}_\theta} \beta_i(0). \quad (3.53)$$

If  $\theta$  is a parameter in an output probability distribution, and  $\mathcal{D}_\theta$  is the set of all  $i \rightarrow j$  that have that output distribution, then

$$\frac{\partial \text{Pr}_m(y_1^T)}{\partial \theta} = \sum_{t=1}^T \sum_{i \rightarrow j \in \mathcal{D}_\theta} \alpha_i(t-1) a_{ij} \left( \frac{\partial b_{ij}(y_t)}{\partial \theta} \right) \beta_j(t). \quad (3.54)$$



In the discrete case, if  $\theta = b_{ij}(k)$ , then

$$\frac{\partial \text{Pr}_m(y_1^T)}{\partial \theta} = \sum_{t: y_t = k} \sum_{i \rightarrow j \in \mathcal{D}_\theta} \alpha_i(t-1) a_{ij} \beta_j(t). \quad (3.55)$$

The derivatives needed in (3.54) for the continuous densities discussed in this thesis are in the appendices. Plugging (3.52), (3.53), and (3.54) into (3.47), we can compute the gradient of the mutual information between  $y_1^T$  and  $m$ , and use it in a gradient-based hill-climbing algorithm.

Notice the similarity between the derivatives of the likelihood function  $\text{Pr}_m(y_1^T)$  and the reestimation formulas in the forward-backward algorithm. In the forward-backward algorithm, we attempt to maximize the likelihood of the training data while maintaining certain constraints on our parameters. For transition probabilities, for example, we can do this by maximizing

$$F(\Theta, \lambda) = \text{Pr}_m(y_1^T) + \sum_i \lambda_i \left( 1 - \sum_j a_{ij} \right), \quad (3.56)$$

as a function of the  $a_{ij}$  in  $\Theta$  and the LaGrange multipliers,  $\lambda_i$ . Setting the derivatives equal to 0, we find that

$$a_{ij} = \lambda_i^{-1} \left( \frac{\partial \text{Pr}_m(y_1^T)}{\partial a_{ij}} \right) a_{ij}. \quad (3.57)$$

If we pretend we know  $a_{ij}$  on the right, we can solve for a new estimate,  $\hat{a}_{ij}$ , on the left. Solving also for  $\lambda_i$ , and substituting into (3.52), we end up with the forward-backward reestimation formula for transition probabilities. So the forward-backward  $\alpha$ - $\beta$  style computation is identical in both MLE, when the forward-backward algorithm is used, and in MMIE, when gradient search is used.

Note that although the computation for each individual model is essentially identical in the two cases, in MLE, we need only do it for the model  $m$  in the training sample to estimate the parameters for the whole close-knit family  $M$ , whereas in MMIE we must perform this forward-backward computation once for every model in the family. So if the family  $M$  is large, the hill-climbing step in MMIE will involve much more computation than is performed in MLE, unless there are certain relationships between the different models in  $M$  which can be exploited.

If we are considering maximizing  $I_\Theta(y_1^T, m)$  by hill-climbing, a natural question to ask is how much computation is required to compute the Hessian of a likelihood,  $\text{Pr}_m(y_1^T)$ ,

which is what is needed to compute the Hessian of  $I_{\Theta}(y_1^T, m)$ . By carrying out the same differentiations here, it can be seen that what is needed to compute this Hessian for an  $n$ -state hidden Markov model is the probability of generating the subsequence  $y_{t1}^{t2}$  while starting in state  $i$  and ending in state  $j$  for all  $1 \leq t1 < t2 \leq T$  and for  $1 \leq i, j \leq n$ , which involves computation of the order  $n^3 t^2$ . For the application we will be considering this amount of computation is not possible.

There are a number of sophisticated hill-climbing techniques which do not use the Hessian, such as conjugate gradient methods, and quasi-Newton methods. In almost all varieties of these methods, a step is taken in the following manner:

1. Compute the gradient.
2. Compute a direction to move in from information in past steps.
3. Perform a line search to optimize the objective function.

The analyses of these techniques almost always assume that the line search is done perfectly. To do a line search involves many computations of the objective function. This is often not that expensive in comparison with a computation of the gradient. In our case, however, to compute the value of the objective function we need to compute all the different  $\alpha$ 's. To compute its derivative we need to compute the  $\alpha$ 's and the  $\beta$ 's. So, it is only twice as much computation to compute the derivative as it is to compute the objective function itself. Another more important way that our case differs from the cases that these standard hill-climbing techniques were designed for is that we can not afford to take nearly as many steps per parameter as normally are taken. In the applications to speech recognition that will be discussed in the following sections, we will see that it involves roughly an hour of mainframe computer time to compute  $I_{\Theta}(y_1^T, m)$ , and we will typically have a few thousand parameters. A quasi-Newton method normally requires a number of steps roughly equal to the dimension of the space to get a good estimate of the Hessian, which is out of the question in our case. As a result of these two peculiarities of our application, we will simply use straightforward gradient descent.

As we iteratively step to new parameter vectors, we must make sure that we maintain whatever constraints there are. As we have seen above, the tying constraints are maintained by summing the partial derivatives of each tied parameter, and then updating tied parameters simultaneously. We can maintain linear constraints, such as

$$\sum_j a_{ij} = 1, \quad (3.58)$$

by projecting the gradient as computed above onto the constraint plane. Inequality con-

straints, such as the requirement that covariance matrices be positive definite and that each transition probability be non-negative can be maintained in two ways. One is to reparameterize the problem in such a way that there are no such constraints. For example, every symmetric positive definite matrix can be rewritten as the square of an ordinary symmetric matrix. The other method applies if the constraining inequalities are linear; it consists of taking a series of steps along the projections of the gradient onto planes which are boundaries of the constraints.

## 4. The IBM Experimental Speech Recognizer

In this chapter, the application of hidden Markov models to speech recognition will be discussed through a description of the experimental speech recognition system which has been developed at the IBM Thomas J. Watson Research Center. This system will serve as the laboratory in which the techniques suggested in this thesis will be tested.

The IBM Continuous Speech Recognition Group has been in existence under the leadership of Fred Jelinek since 1972. Its approach to speech recognition has been based on hidden Markov models from the start. The system which this group has developed is a natural-language speaker-dependent system. In 1981, a decision was made to implement a real-time recognizer. In order to do this with the computational resources then available, the task, which had always been continuous-speech recognition, was shifted to isolated-word recognition. Although this shift in task was made, the system is still based on continuous-speech techniques. The current real-time system has a vocabulary of 20,000 words, and a per-word recognition accuracy of roughly 95 percent.

Speech is captured by a Crown PZM-6S pressure-zone microphone, and sampled at 20,000 Hertz with a 12 bit A/D converter. Every centisecond a 512-point FFT is applied to the Hamming-windowed digital waveform. A critical band filter bank is simulated by taking linear combinations of the 256 energies resulting from the FFT. There are twenty filters which are placed linearly from 250 and 1000 Hertz, and logarithmically from 1000 to 7500 Hertz.

Long-term estimates of signal and noise are kept, and log energies from the filters are individually normalized to have constant long term levels and dynamic ranges. The next signal-processing step is the application of a model which attempts to model the hair cells in the human ear. Log energies are processed so that ear model filter outputs decrease in steady-state regions and peak sharply at transitions [Cohen 87]. The final signal-processing step is vector quantization. Each centisecond the 20-parameter ear-model output is compared to each of 200 templates using a Euclidean distance measure. The name of the closest template becomes the output of the signal processing during that centisecond. The 200 speaker-dependent templates are constructed automatically by K-means clustering [Nadas 81]. Each codeword is referred to as an *acoustic label*.

The 100 bytes of acoustic labels output each second by this signal-processing module are input to an acoustic-modeling module which is based on hidden Markov models. Each word is composed of a sequence of phones, each of which is associated with a hidden Markov

model. Each phone model has a single initial and a single final state. A word model is constructed from a sequence of phone models by making the initial state of the first phone model the initial state of the word model, making the final state of the last phone model the final state of the word model, and by creating null transitions from the final state of the  $i$ th phone model to the initial state of the  $(i + 1)$ th phone model. In the same way, hidden Markov models for words can be concatenated to form a single hidden Markov model for a sequence of words. More sophisticated methods of combining small models into large ones are also used. Phonetic rules governing the coarticulation of sounds can be modeled by using phone models with multiple initial and multiple final states, and by connecting adjacent phone models with null transitions in a manner which depends on the identities of the phones as well as other nearby phones.

Each non-null transition in a phone model is labeled with an output-distribution number. Transitions that are labeled with the same output distribution number have tied output distributions. The transition labels in a word model are inherited from the phones in the word. Therefore, in the model for a word like *Mississippi* which contains more than one instance of a phone, there will certainly be transitions with tied output distributions. Transition-probability distributions associated with states in different instances of the same phone are also tied in a similar fashion. Typically, the system uses 1000 distinct transition probabilities and 200 different output distributions. Each output distribution is a discrete distribution over the 200 different acoustic labels used in the vector quantization step. The system, thus, has on the order of 40,000 parameters in its acoustic models.

These 40,000 parameters are trained with the forward-backward algorithm, using a training sample consisting of 100 naturally occurring sentences, which can be spoken in about 20 minutes. Normally, three iterations of the forward-backward algorithm are used, followed by a fourth smoothing iteration that is designed to smooth out the parameter estimates in an effort to avoid biasing them too strongly towards peculiarities in the training sample [Jelinek 80]. This smoothing process can, in fact, be seen as an instance of empirical Bayesian estimation, in which maximum likelihood estimates are converted to maximum *a posteriori* estimates using an empirical prior.

The prior probability of a word sequence  $w_1^T$  is computed with a language model which is founded on the following assumption:

$$\Pr(W_t = w_t \mid W_1^{t-1} = w_1^{t-1}) = \Pr(W_t = w_t \mid W_{t-2} = w_{t-2}, W_{t-1} = w_{t-1}). \quad (4.1)$$

Using this assumption, and leaving out the random variables for brevity,

$$\Pr(w_1^T) = \Pr(w_1) \Pr(w_2 | w_1) \prod_{t=3}^T \Pr(w_t | w_{t-2}, w_{t-1}). \quad (4.2)$$

The transition probabilities in this second-order Markov model are estimated from a sample of 25 million words of text using a scheme described by Katz. [Katz 87].

Recognition is performed using a stack search algorithm [Bahl 83]. The search relies on two basic components: a *fast match*, which provides a list of candidate words that can begin at a given point of time, and a *detailed match*, which evaluates each candidate word in detail. This thesis is concerned almost entirely with the detailed match.

## 5. Continuous Parameters

### 5.1 The Quantization Problem

The information entering a recognizer is the acoustic waveform picked up by the recognizer's microphone. Ideally, we would like the recognizer, as a whole, to use models which permit it to extract as much information as possible from this input waveform about the corresponding word sequence. The analog waveform, however, is not modeled directly. It is first digitized so that it can be manipulated with a digital computer. Because the resultant digital signal can be reconverted to an analog signal which is indistinguishable to the human ear from the original signal, we can assume that little, if any, information relevant to the recognition of the speaker's words is lost by this step. Note, however, that had the analog signal been sampled at, say, 4,000 Hertz with a 4-bit A/D converter, there would have been a noticeable difference between the original and the reconstructed acoustic signals. Furthermore, it is likely that a person would have some difficulty recognizing reconstructed speech, because certain relevant information would have been lost. What determines the sampling rate and word size of the A/D converter is the principle that no relevant information should be discarded by the conversion. This principle, in fact, applies to the signal processor as a whole. By the data-processing theorem, the sequence which is output by the signal processor contains no more information about the spoken word sequence than exists in the acoustic waveform which is received by the microphone; signal processing can only destroy information [Gallager 68]. The obvious question, then, is why do any signal processing at all, beyond converting the waveform to a form which can be processed by a digital computer. In Chapter 9, we will, in fact, describe an attempt to model the digitized waveform directly. However, we will find that the results of this attempt are discouraging, and will be forced to conclude that we do not, at the present time, know of a family of statistical models with a relatively small number of parameters that is capable of directly modeling speech waveforms accurately. The goal of the signal processing module is to convert the acoustic waveform into a form that we do have some idea of how to model, while, at the same time, destroying as little information as possible. A model of the signal-processed output will then indirectly serve as a model of the original waveform.

After the waveform has been digitized, the next sequence of steps in the IBM system is, in essence, to extract the energy in 20 filters every 10 milliseconds. The placement and the

width of the filters is determined by the principle of not discarding information about the word sequence. Thus, it is deemed relatively unimportant whether there is energy at 6900 Hertz as opposed to at 7000 Hertz. Similarly, because human articulators have inertia, it is assumed that the transfer function of the vocal tract does not change rapidly, and that no important information will be lost by assuming that the signal is steady-state over a one-centisecond interval.

The final step in the signal processing is vector quantization. This step makes it possible to use nonparametric, discrete output distributions, which cannot be inherently inaccurate because they assume nothing. The disadvantage to this step is that there may be some information lost. The amount of information lost can be made negligible by using a large enough number of codewords in the vector quantizer. As the number of codewords increases, so does the amount of training data needed to reliably estimate the output distributions. If the training data is of limited size, then the size of the codebook used by the vector quantizer must also be of limited size. But with a small codebook there may be some information lost.

An alternative to having to make the best out of this trade-off, is to use the 20-parameter vectors that are the output of the ear model and the input to the vector quantizer directly. By doing this, we can be sure that we do not lose any information by vector quantization. To make use of the additional information in the 20-parameter vectors, however, we must use output densities which are accurate.

There have been a number of speech recognition systems built which use continuous-parameter hidden Markov models. Recently, for example, Poritz and Richter have reported significant performance improvements using such models [Poritz 86]. There have also been a number of previous attempts to remove the vector quantization from the IBM system [Bahl 81]. But despite the success of others working on other tasks with other systems, all these previous attempts to incorporate continuous-parameter models into the IBM system have failed; the performance of the system has always deteriorated. It must have been that, either the families of output densities were inaccurate, or they had too many free parameters. The danger of using inaccurate densities was particularly acute because in every attempt, parameters were estimated with MLE, which, as we have seen, derives its justification from an implicit assumption of model correctness. This is, then, an ideal situation in which we would hope to benefit from the application of MMIE which does not presume model correctness. In subsequent chapters, we will see that the performance of the IBM system can, in fact, be improved by using MMIE in conjunction with continuous parameters.



## 5.2 Output Densities

The obvious first candidate for a family of output distributions is the family of multivariate Gaussians, since 1) by the central limit theorem, the distribution of the sum of a large number of independent random variables tends towards a Gaussian, 2) a Gaussian distribution has the greatest entropy of any distribution with a given variance, and 3) if some random variable has a Gaussian distribution then so does any linear transformation of that variable.

An  $n$ -dimensional multivariate Gaussian density with mean,  $\mu$ , and inverse covariance matrix  $W$  has the following form

$$N(\mathbf{x} : \mu, W) = \frac{|W|^{1/2}}{(2\pi)^{n/2}} e^{-\frac{1}{2}(\mathbf{x}-\mu)'W(\mathbf{x}-\mu)}. \quad (5.1)$$

Notice, that 1) the number of steps to compute the density at a point for an  $n$ -dimensional Gaussian increases as the square of  $n$ , 2) the density is unimodal, with the mode at the mean,  $\mu$ , and 3) the probability density at a point drops off exponentially in the square of the distance of the point from the mean.

Typically, a speech recognition system will use 200 20-dimensional output distributions, and typically, a density at each one of these distributions will have to be evaluated 100 times a second. If full-covariance Gaussians are used, the system will have to perform roughly 4 million multiply-adds per second just to evaluate output densities. While not a prohibitive amount of computation, it is certainly not negligible. One way to drastically reduce this computation is to assume that the off-diagonal terms in the covariance matrices, and, as a result, in the inverse covariance matrices, are 0. Using the same typical numbers, we find that a system which uses such diagonal Gaussian output distributions will only require 400,000 multiply-adds per second to compute output densities. This is a reduction by a factor of 10 in the amount of computation required for full-covariance Gaussians. Another advantage to using diagonal Gaussians is that the number parameters to be estimated is also reduced by a factor of 10, which means that less training data and time will be required. The disadvantage is that the assumption that different elements of the observation vector are uncorrelated may be so inaccurate as to significantly degrade a system's performance. The extent to which this is the case clearly depends on the signal processing and, as we shall see, on the word models used in a particular system. It is therefore an empirical issue as to whether the computational expense of a full-covariance Gaussian is worth the performance improvement, if any. Results of experiments comparing the performance of different output

distributions to one another are presented in Chapter 11.

It is worth noting that a similar computation-versus-modeling-accuracy trade-off can occur in a system which uses discrete output distributions. In the IBM system, for example, vector quantization is performed with a Euclidean distance metric. If observation parameters are correlated with one another, this metric will be inappropriate. Yet it is used precisely because a decision has been made that modeling the correlation is not worth the additional computation and training data.

Another point worth mentioning here is that if we have prior information about our parameters, then it may be possible to use models which take advantage of this information. For example, suppose our parameters were the energies in different spectral bands. We would expect energies in adjacent bands to be more correlated with one another than energies in bands which are not adjacent. Let  $\rho(i, j)$  be the correlation between the energy in band  $i$  and the energy in band  $j$  for a given distribution. We might be able to adequately model the structure of a covariance matrix of these parameters by assuming that  $\rho(i, i+2) = \rho(i, i+1)\rho(i+1, i+2)$ . Such a model has many fewer parameters than a full-covariance model, but may be almost as accurate.

The extent to which unimodal output densities in a speech-recognition system are adequate depends on both the signal processing in the system, and on the shapes of the word models used by the system. Consider a speaker-independent system, for example. Because the vocal tracts of women are normally shorter than the vocal tracts of men, the formant frequencies for a given sound will tend to be higher in a woman's voice than in a man's. One can imagine, then, that the distributions of energy at a given frequency for a particular sound will be bimodal: one mode for men, the other for women. But, if the signal processor successfully performs some sort of long-term speaker normalisation, this difference between men's and women's voices will be removed, and unimodal models will be accurate. Alternatively, multimodal distributions can be modeled as mixtures of unimodal distributions and, as we have seen, output distributions which are mixtures of other output distributions can be modeled by using hidden Markov models with multiple paths between states. Consider the word *the*, which can be pronounced with either a short or long vowel. Consider the two possible word models for *the* in Figures 5.1 and 5.2. In the simple model, unimodal output densities on the vowel transitions would surely be inadequate. In the complicated model, however, they may be perfectly adequate, since the different pronunciations of the vowel are modeled by different paths. Note that in the discrete case, it makes no difference which model is used since a mixture of non-parametric

discrete distributions is also discrete. From this example, we can see, then, that more care must be paid to the design of word models when using parametric continuous-parameter output distributions than when using nonparametric discrete output distributions.

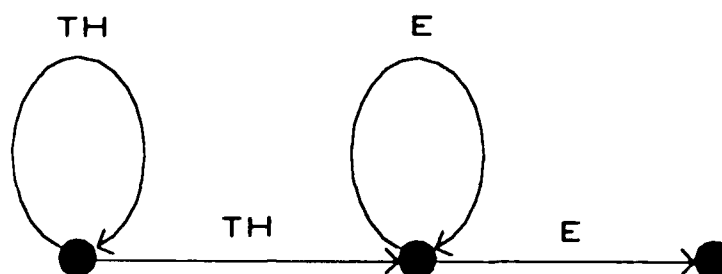


Figure 5.1: Simple Model for the

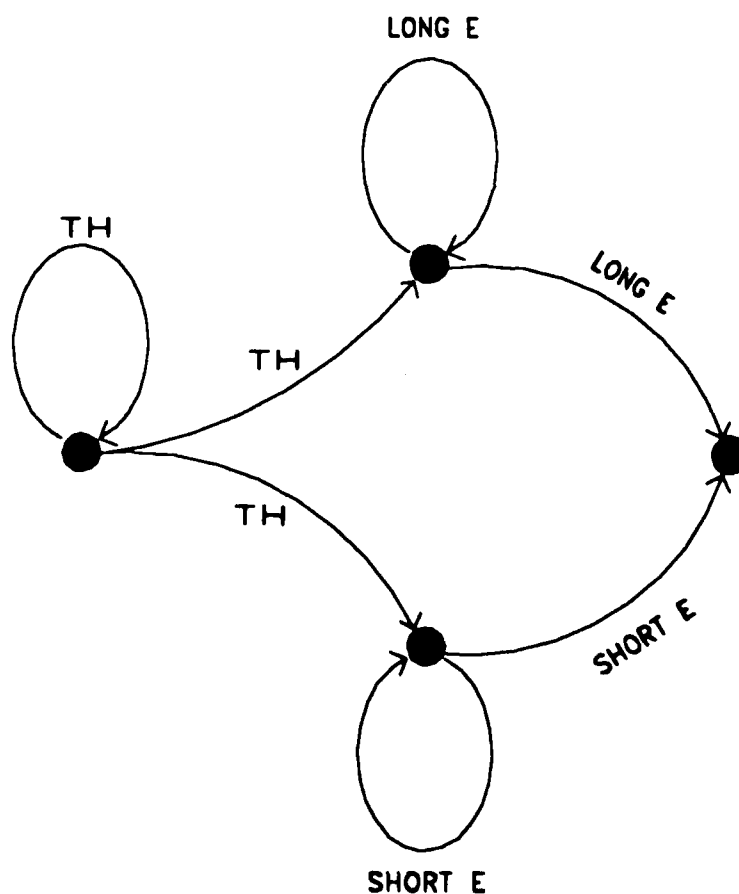


Figure 5.2: Complicated Model for the

The next point we observed regarding Gaussian densities is that the probability density drops off exponentially in the square of the distance from the mean. This means that a Gaussian is not particularly robust to outliers. A spurious sample can result in an extremely small output probability. As a result, outliers in the training data will tend to have a relatively large effect on parameter estimates, and outliers in the test data will tend to have a relatively large effect on acoustic model probabilities. The robustness problem and various models which are robust to outliers are described by Peter Huber [Huber 81].

One approach to this robustness problem is to use a mixture of Gaussians. For example, there could be one with a relatively small variance, which models most samples, and another with a relatively large variance, which models a small number of outliers. The problem with this approach is that a mixture of two Gaussians contains more than twice as many parameters as are in a single Gaussian of the same dimension. If we have more parameters, we will need more training data. Another drawback is that it involves twice as much computation to compute the probability density when using a mixture of two Gaussians as it does when using a single Gaussian.

At a recent conference, Alan Richter suggested using a mixture of Gaussians in which the means of the individual Gaussians are constrained to be identical, and in which the covariance matrices are constrained to be multiples of one another [Richter 86]. I will refer to such a mixture as a *Richter model*. The density of a Richter model is of the form

$$R(\mathbf{x} : \mu, W, m, c, \lambda) = \sum_{i=1}^m \lambda_i N\left(\mathbf{x} : \mu, \frac{1}{c_i} W\right). \quad (5.2)$$

Because the quadratic forms in each of the individual Gaussians are just multiples of one another, the amount of computation of the density at a point using a Richter model is essentially the same as the amount of computation using a single Gaussian. The number of free parameters in a Richter mixture of  $m$  Gaussians is only  $2m - 2$  greater than the number of free parameters in a single Gaussian.

In figure 5.3, the solid line depicts the density of a Richter model with  $m = 3$ ,  $c_1 = 1$ ,  $c_2 = 2$ ,  $c_3 = 4$ ,  $\lambda_1 = .773$ ,  $\lambda_2 = .191$ , and  $\lambda_3 = .036$ . The  $c$ 's were held fixed, while the  $\lambda$ 's were estimated using MLE on actual speech data. The dashed line depicts the density of an ordinary Gaussian which has parameters that were estimated from the same speech data. The horizontal axis is scaled in standard deviations from the mean. Clearly, the densities are extremely close to one another. In the plot of the log densities of the same two distributions, it is obvious that the tails of the Richter model drop off much more slowly than those of the ordinary Gaussian. So we can see that for this data the Richter model is

extremely close to an ordinary Gaussian except in the tails. By using such a mixture we can have all the benefits of a Gaussian and, in addition, have a model which is significantly more robust to outliers.

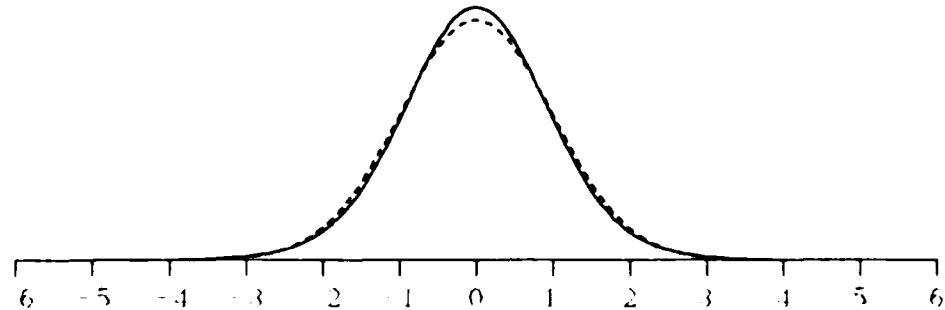


Figure 5.3: Richter (solid line) vs. Gaussian Density (dashed line)

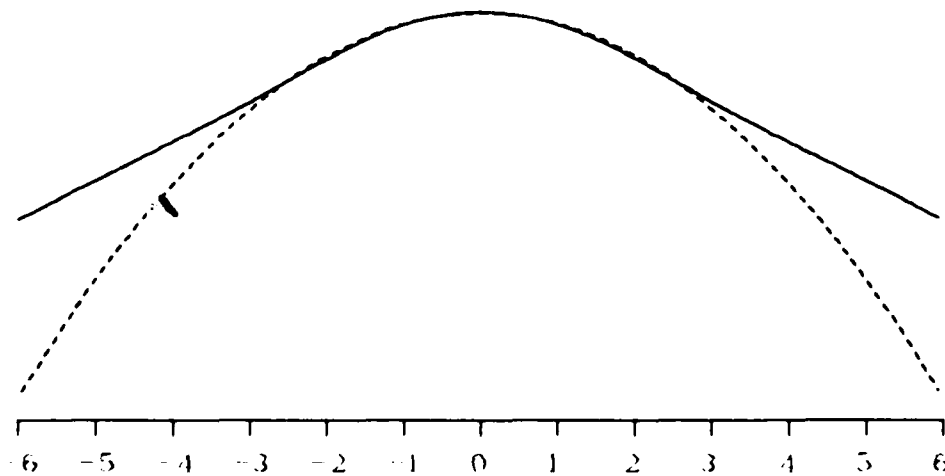


Figure 5.4: Richter (solid line) vs. Gaussian Log Density (dashed line)

Parameters for a hidden Markov model with Richter output densities can be estimated by treating a single transition with a Richter distribution as a collection of  $m$  parallel transitions with individual Gaussian output distributions. The forward-backward algorithm is then run in the usual way except that during the reestimation step, it is necessary to impose the constraint that the individual Gaussians must have the same means, and

variances which are multiples of one another. The MLE reestimation formulas along with the derivatives necessary for MMIE hill-climbing for hidden Markov models with Richter output distributions can be found in Appendix 13.4.

### 5.3 Mixtures

There is a problem with estimating all the parameters in a Richter model with MLE. Consider a Richter mixture of two Gaussians which is to be trained from the sample  $y_1^T$ . Suppose that  $\lambda_1 > 0$ , and, for some  $i$ ,  $1 \leq i \leq T$ ,  $\mu = y_i$ . As  $c_1 \rightarrow 0$ , the probability density at  $y_i$  will go to infinity. If  $c_2 > 0$ ,  $\lambda_2 > 0$ , and  $|W^{-1}| > 0$ , then the probability density at every other sample point will be greater than 0. So the probability density at  $y_1^T$  will be infinite, even if  $\mu = y_i$  is a point which is wildly different from all the other sample points. In order to avoid this problem, we will simply leave the  $c_i$ 's fixed and not reestimate them.

This problem is not restricted to Richter models. It can occur in many kinds of mixtures. Consider a hidden Markov model with one state and two self loops with different output distributions. Again, if the probability density at one sample point is infinite for one of the distributions, and the density of every other point is non-zero for the other distribution, then the density of the training data as a whole will be infinite.

As another example, consider the problem of estimating the parameters of the hidden Markov model shown in Figure 5.5. If distribution  $A$  has infinite density at the first sample point,  $y_1$ , and  $B$  has non-zero density at  $y_2^T$ , the density at  $y_1^T$  will be infinite.

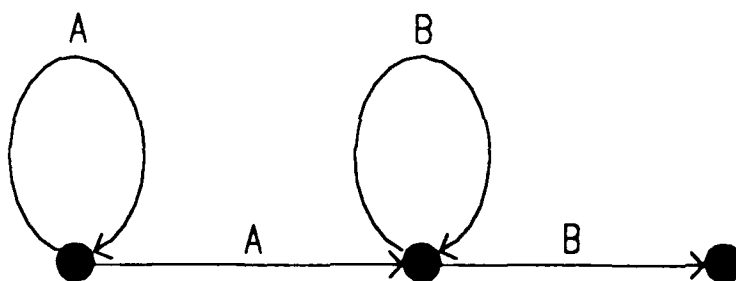


Figure 5.5: Simple Model with Mixture Problem for MLE

This problem, has the potential of occurring in any hidden Markov model in which it is possible to account for a single  $y_i$  with one distribution and all other  $y_j$ 's with other distributions. Unfortunately most hidden Markov models that are used for speech recognition are of this nature.

One way to avoid these ugly estimates is to impose artificial limits on the parameters of the output distributions. Limiting variances is the obvious example of this. What is often done however is to simply ignore the problem. This tends to work out all right because the forward-backward algorithm is usually not run for enough iterations to find one of these nasty points. Nonetheless, this is all very unsettling. On the one hand, the forward-backward algorithm is used because it attempts to maximize the likelihood of the training data, while, on the other hand, one must hope it never really succeeds.

Intuitively, we feel that it is ridiculous to use a whole subdistribution to model one and only one sample point when estimating the parameters of a mixture of distributions. Why is this? After all, such an estimate does maximize the likelihood of the sample.

One reason for this strong intuition is that if we move the one sample point by a small amount, one entire subdistribution will no longer contribute to the probability of generating the training sample, and the probability density of the training sample will drop dramatically. When we use parametric densities like Gaussians, we assume that the probability density of a sample will be a continuous function of the sample, and therefore that a small perturbation to the sample will result in a small perturbation to the probability density.

In order to understand another more important reason why such estimates seem ridiculous, let us examine the purpose of making any estimates in the first place. In our case, we estimate parameters in order to classify acoustic signals as being associated with one word or another. If we use an entire subdistribution to model one sample point which represents a centisecond example of an acoustic signal, it is likely that we will do a very good job of classifying a section of the acoustic signal around that centisecond. But unless there are other examples of the exact same acoustic parameter values in the training sample, that subdistribution will be of no use in classifying any other section of the acoustic signal. One subdistribution will have been used to classify a small section of the training data extremely well, but will not help at all in classifying the rest of the data. Normally the number of words represented in the training data will be significantly greater than the number of subdistributions in the collection of models. For the purpose of speech recognition, then, it is a bad idea to use an estimate which uses an entire subdistribution to model one and only one centisecond sample because although such an estimate will provide a lot of information about one word or phrase, it will provide relatively little information about the whole sequence of words which is the training script.

This reasoning suggests that if we use MMIE we will not encounter this problem with

mixtures because the objective will be to maximize the information provided about the whole training script. The problem with MLE is that it is possible to make the likelihood of any one sample point infinite. But with MMIE, it is not possible to provide more information about a word than minus the log of the prior probability of that word. Unless either the per-word entropy of the recognizer's language is very large, or the number of words in the training sample is very small relative to the number of subdistributions in the collection of acoustic models, MMIE should estimate the parameters of mixtures in such a way that the resultant models will be useful in recognizing new speech. This is very comforting since almost all the continuous-parameter hidden Markov models used in speech recognition are mixtures of one form or another.



## 6. Parameter Reduction

### 6.1 The Parameter Reduction Problem

As we have discussed in previous sections, there is an important trade-off between the complexity of an acoustic model and the quality of the estimates of the parameters in that model. On the one hand, the greater the number of parameters in a model, the greater the potential the model will have to represent complex acoustic events. On the other hand, the greater the number of parameters in a model, the more variance there will be in the estimates of the probabilities of these acoustic events.

Let us examine this problem in the context of the dimension of the acoustic-observation vectors produced by the signal-processing component of a speech-recognition system. Assume we have a fixed number of acoustic vectors of training data. Suppose we are using diagonal Gaussian output distributions. The log of the probability density in a distribution with mean vector  $\mu$  and standard deviation vector  $\sigma$  at an  $n$ -dimensional acoustic observation vector  $y$  is

$$\log b(y) = -\frac{1}{2} \sum_{k=1}^n \left[ \log(2\pi\sigma_k^2) + \left( \frac{y_k - \mu_k}{\sigma_k} \right)^2 \right]. \quad (6.1)$$

Assuming that the variance of the estimates of the  $\mu_k$ 's and  $\sigma_k$ 's are independent of  $k$  and  $n$ , the variance of  $\log b(y)$  will increase linearly with  $n$ , the dimension of the acoustic observation vectors. If the output distribution was a full-covariance Gaussian, the variance of the log of the density estimate would increase with the square of the dimension.

We would like the signal processing to communicate fine details of the acoustic spectrum every centisecond, so that we can accurately estimate the probability that the sound in a particular centisecond was generated from a particular phone. But if the signal processing represents this detail in a large number of parameters, we will be stuck with large variances in our estimates of these probabilities.

Ideally, we would like the signal processing to output only a small number of parameters every centisecond, and we would also like these parameters to be as informative as possible about what sound is being spoken. One way to move towards this goal is to apply a transformation which maps the signal-processing output to a smaller dimension while preserving as much information as possible about the sound which has been spoken.

## 6.2 Linear Discriminants

Let us consider the parameter reduction problem in a general setting in which we have a collection of input vectors and associated class labels. In speech recognition the input vectors are the vectors output by the signal processor every centisecond, and the associated classes may be, for example, the phones that the speaker is presumed to have uttered. We would like to find a transformation which maps input vectors to output vectors of lower dimension and which preserves as much information as possible about the classes associated with the input vectors. For simplicity we will restrict ourselves to linear transformations.

The method most commonly used for this purpose is principal component analysis. The first principal component of a sample vector is the projection of that sample onto the direction along which there is the largest variance over all samples. The rationale behind the use of principal components is an assumption that the direction along which there is the most variation is likely to contain the most information about the classes associated with the input vectors. Because we want the  $n$ th principal component to contain information which is not already found in the first  $n - 1$  components, we select the  $n$ th component to be the linear combination which has the largest variance given the constraint that it must be uncorrelated with the first  $n - 1$  components. We will see, below, that the  $n$ th principal component is the eigenvector with the  $n$ th largest eigenvalue of the overall covariance matrix of the signal-processor output vectors.

One problem with principal components is that they are not independent of parameter scale. If we multiply the  $n$ th parameter of each input vector by some constant, it cannot possibly effect the amount of information contained in the  $n$ th parameter about the classes which are associated with the input vectors. But, as we multiply the  $n$ th parameter by larger and larger values the variance of this parameter will become larger and larger and the first principal component will approach this scaled-up  $n$ th parameter. For this reason, when principal components are used in pattern recognition the input parameters are normally scaled to have unit variance. This trick makes the resulting principal components invariant to linear transformations of the input samples by diagonal matrices. Unfortunately, it does not make them invariant to arbitrary full-rank linear transformations.

Figure 6.1 depicts another problem with using principal components in pattern recognition. Suppose our data consists of samples from two 2-dimensional Gaussian classes, and that the covariance matrices for the two Gaussians are identical. The ellipses represent contours of equal probability density for the two Gaussians. The solid line is in the direction

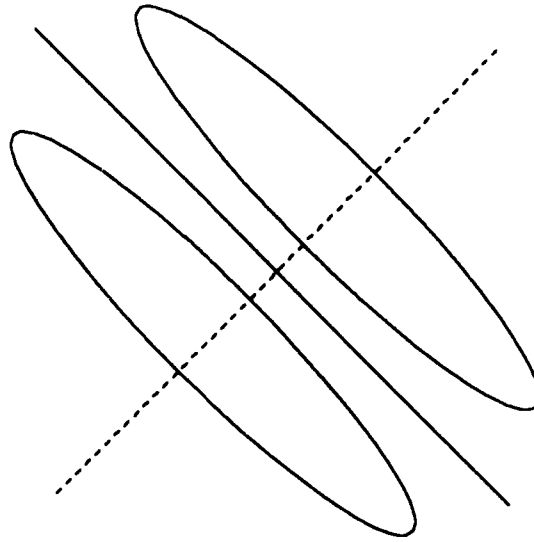


Figure 6.1: Two Gaussian Classes

of maximum variance for each of the Gaussians. In this example it is also in the direction of maximum variance for the mixture of the two Gaussians, and therefore in the direction of the first principal component. But a projection onto this line contains no information as to which class a sample is likely to belong. The most information can be found in a projection along the dashed line which is perpendicular to the first principal component. The problem is that it is not necessarily true that the direction of greatest overall variation contains the most information about class membership. In particular, the variations within the individual classes must also be taken into account.

One way of attacking both of these problems with principal components is to choose projections which maximize the ratio of the overall variance to the average within-class variance [Wilks 61] [Friedman 67]. I shall refer to the projections which maximize this ratio as *principal discriminants*. Any full-rank linear transformation of the input samples will appear in both the numerator and the denominator of this ratio, and will divide out, leaving the ratio unaffected. Principal discriminants are thus invariant to all full-rank linear transformations of the input. Now reconsider figure 6.1. Since the means of the two classes have the same projection onto the solid line, the overall variance in the direction of the solid line is equal to the average within-class variance in that direction. This means that the ratio of total to average variance of projections onto the solid line will have the smallest possible value, 1, which is what we want since, as we noted above, there is no information along

this line. It has, in fact, been shown by Nadas [Mercer 86] that under certain Gaussian assumptions the projections which maximize this ratio are also those which maximize the average mutual information between the projections and the classes.

Suppose we have a sample of  $n$   $p$ -dimensional elements, with  $n_i$  elements in class  $i$ . Assume that this sample has zero mean. In practice, we can ensure that this is the case by subtracting off the mean from the sample elements. Let  $T$  be the covariance matrix of the entire sample, and  $W_i$  be the covariance matrix of the  $i$ th class. Let  $W$  be the average within-class covariance matrix

$$W = \frac{1}{n} \sum_i n_i W_i. \quad (6.2)$$

The variance of the projection of the sample onto a vector  $\mathbf{v}$  is  $\mathbf{v}'T\mathbf{v}$ . We would like to find a vector  $\mathbf{v}$  which maximizes

$$\frac{\mathbf{v}'T\mathbf{v}}{\mathbf{v}'W\mathbf{v}}. \quad (6.3)$$

We can do this by maximizing  $\mathbf{v}'T\mathbf{v}$  while constraining  $\mathbf{v}'W\mathbf{v}$  to be equal to some constant, such as 1. Using the LaGrange multiplier  $\lambda$ , we want to choose  $\mathbf{v}$  to maximize

$$f(\mathbf{v}, \lambda) = \mathbf{v}'T\mathbf{v} - \lambda(\mathbf{v}'W\mathbf{v} - 1). \quad (6.4)$$

This amounts to choosing  $\mathbf{v}$  and  $\lambda$  such that

$$\mathbf{v}'W\mathbf{v} = 1, \quad (6.5)$$

and

$$T\mathbf{v} = \lambda W\mathbf{v}. \quad (6.6)$$

Equation (6.6) is the general eigenvector equation, and is solved by  $p$  eigenvectors with real eigenvalues, due to the symmetry of  $T$  and  $W$ . The solution which maximises  $\mathbf{v}'T\mathbf{v}$  is the eigenvector with the largest eigenvalue. This is because the variance of a projection onto an eigenvector  $\mathbf{v}$  is

$$\mathbf{v}'T\mathbf{v} = \mathbf{v}'\lambda W\mathbf{v} = \lambda. \quad (6.7)$$

Let  $\mathbf{v}_1$  and  $\mathbf{v}_2$  be eigenvectors with eigenvalues  $\lambda_1$  and  $\lambda_2$ . We have

$$\lambda_2 \mathbf{v}_1' W \mathbf{v}_2 = \mathbf{v}_1' T \mathbf{v}_2 = \mathbf{v}_2' T \mathbf{v}_1 = \lambda_1 \mathbf{v}_2' W \mathbf{v}_1 = \lambda_1 \mathbf{v}_1' W \mathbf{v}_2. \quad (6.8)$$

Assuming that  $\lambda_1 \neq \lambda_2$ ,

$$\mathbf{v}_1' W \mathbf{v}_2 = 0, \quad (6.9)$$

and

$$\mathbf{v}_1' T \mathbf{v}_2 = \lambda_2 \mathbf{v}_1' W \mathbf{v}_2 = 0. \quad (6.10)$$

So the projection of the sample onto  $\mathbf{v}_1$  is uncorrelated with its projection onto  $\mathbf{v}_2$ . Let  $\mathbf{v}_n$  be the eigenvector with the  $n$ th largest eigenvalue, and normalize  $\mathbf{v}_n$  such that  $\mathbf{v}_n' W \mathbf{v}_n = 1$ . Then  $\mathbf{v}_n$  maximizes  $\mathbf{v}_n' T \mathbf{v}_n$  subject to the constraint that  $\mathbf{v}_n' W \mathbf{v}_n = 1$ , and to the constraint that the sample's projection onto  $\mathbf{v}_n$  be uncorrelated with its projection onto  $\mathbf{v}_i$ , where  $i < n$ . We can define  $\mathbf{v}_n$ , then, to be the  $n$ th principal discriminant.

If we wish to reduce the number of dimensions of our sample from  $p$  to  $q$  using linear discriminants, we project the  $p$ -dimensional sample elements onto the first  $q$  principal discriminants. If we have  $k$  classes, then all but the top  $k - 1$  eigenvalues will be equal to 1, since any contribution to  $T$  not included in  $W$  must come from variance of the class means which lie in a  $k - 1$  dimensional plane. Therefore, if we are reducing data from  $k$  classes down to  $q$  dimensions,  $q$  must be less than  $k$ .

After having projected the sample data onto a set of principal discriminants, the off-diagonal terms of the sample covariance matrix will be 0. We might, therefore, also expect the off-diagonal terms of the sample covariance matrices for the different classes to be relatively small. This will tend to be the case if the original sample class covariance matrices are very similar to one another, but will not be the case if they are not. The relevance of this point is that it may be that after the data has been projected onto principal components or principal discriminants, the different class distributions can be modeled adequately with diagonal Gaussians.

Another point worth mentioning here is that after projecting our original parameter vectors onto linear subspaces, we do lose information regarding which parameter is what physical measurement. But, as long as we are using output distributions, such as a full-covariance Gaussians, which do not take advantage of prior information about the peculiarities of our original parameters, the loss of this information will not matter.

Recognition results with different output distributions and various numbers of parameters are presented in Chapter 11.

### 6.3 Application to Speech Recognition

As was described above, our problem is to reduce the dimensionality of the parameter vectors output by the signal processor, while at the same time preserving as much information as possible about the sounds which have been spoken.

In order to apply linear discriminants to this problem, we would like to know which sounds correspond to which acoustic parameter vectors, so that we can compute the covariance matrices for the different sounds. Because we know the training script, we know what words were spoken while the acoustic data in our training sample was produced, but we do not know which sections of the acoustic data correspond to which sounds. If we think of each sound that is of interest to us as being represented by an output distribution, and if we think of the acoustic data as being generated by the hidden Markov model corresponding to the training script, then the forward-backward algorithm will allocate a section of speech to an output distribution according to the probability that that section of speech was generated using that output distribution, and, by so doing, will enable us to compute the covariance matrices we need. In fact, if we are using full-covariance output distributions, then the output distributions computed in the final pass of the forward-backward algorithm will contain these covariance matrices.

We begin, then, by using the forward-backward algorithm to compute covariance matrices for the different sounds in our models. We then use these matrices and the covariance matrix of the whole training sample to compute a set of principal discriminants. Finally, we add a step to our signal processor which projects the original signal processor output vectors onto a subset of these principal discriminants. We can then proceed with these new vectors in the usual way using whatever estimation and recognition scheme we would otherwise use.

One minor problem with this scheme is that some sounds occur much more frequently than others. The sound representing background noise, for example, occurs very frequently, particularly in isolated speech. As a result, the average within-class covariance matrix will be dominated by the covariance matrices of a few output distributions. It might be argued that this is as it should be because it is more important to discriminate between sounds that occur more frequently. But it is not clear that long lasting sounds, like vowels, are more important than consonants in discriminating between different words. In isolated speech, for example, there is usually as much silence as there is speech, and, as a result, the contribution to the average within-class covariance matrix from the sound representing background noise will be much greater than that from any other sound. Yet, this distribution for the most part only helps determine where words begin and end; it does not play a large role in discriminating one word from another. Although one can no doubt improve the basic principal discriminant scheme by weighting sounds according to their importance in discriminating words from other words, in the work described in this thesis I will only do

the bare minimum in confronting this problem by excluding background noise from both the total and average within-class covariance matrices. A more powerful approach which biases the components so that they discriminate between certain pairs of sounds more than others is described by Brown [Brown 84].

There is another minor problem with the scheme outlined above. We want to use principal discriminants in order to reduce the dimensionality of the acoustic vectors so that we can better estimate the parameters in the output distributions in our hidden Markov models. To do this we use the covariance matrices from output distributions that are estimated using our original parameters. But it is precisely because we don't trust these estimates that we want to compute principal discriminants in the first place. This problem can be attacked by stepping down from a large dimension to a small dimension gradually. Although, initially, we may not have particularly good estimates of the covariance matrices for the different sounds, it is still likely that the few components which have the lowest eigenvalues can be safely discarded. After doing this, we can then use the forward-backward algorithm again to estimate covariance matrices for the new acoustic data which has vectors of slightly smaller dimension. These covariance matrices will be somewhat more accurate than the previous ones, and we can again hope to safely discard the few components with the smallest eigenvalues. We can continue in this fashion until arriving at our desired dimension.

The question of what is the best dimension to reduce the acoustic parameter vectors to must be addressed on a case-by-case basis. The answer will depend, among other things, on the number of output distributions, the type of output distributions, the amount of training data available, the amount of computer power available, and on the importance of achieving different performance rates.

## 7. Adjoining Acoustic Parameter Vectors

Certain acoustic events in speech are distinguished by phenomena that occur over time. Consider, for example, those phones which are referred to as *labials* because they are produced with the lips. The acoustic signatures of these phones can be very similar to those of other phones. It is, for example, often difficult to distinguish the labials *b*, *m* and *p* from the alveolars *d*, *n* and *t*. An important property of a labial is that it often pulls down the formants of the sounds surrounding it. Thus, the formants of a sound preceding a labial tend to descend in frequency going into it, and the formants of a sound following a labial tend to rise in frequency coming out of it. An important clue, then, to the presence of a labial is found in information of how the acoustic spectrum is changing in time.

In many systems, including IBM's, the parameters output by the signal processing reflect some function of the energy in different frequency bands during a short interval of time. There is virtually no information about how this short-term spectrum is changing in time. The only way for such a system to model time-derivative information at some time  $t$  is for it to remember the spectrum from time  $t - 1$ . In a system which uses hidden Markov models, this spectrum must somehow be represented by the state that the underlying Markov chain is in at time  $t - 1$ . The problem with this is that the spectrum of the preceding sound might vary considerably. If the system is to remember all the different variations it must have a tremendous number of states and output distributions. So, for example, in figure 7.1 the simple word model for the word *am* on the left would have to be converted to a more complicated model like the one on the right. If the word models are made significantly more complicated in this way, to model this derivative information properly they will need to contain significantly more parameters, which will significantly increase the variance of the probability estimates that are made from them.

An alternative to remembering the spectrum from the previous time in the state of the Markov chain is simply to include it in the parameter vector that is passed from the signal-processing component of the system to the acoustic-modeling component. The original acoustic parameter sequence,  $y_1, y_2, \dots, y_T$ , would thus be converted to the sequence  $y_1 y_2, y_2 y_3, \dots, y_{T-1} y_T$ , by adjoining the spectra from adjacent frames. Time-derivative information could then be modeled directly with multivariate output distributions. Alan Poritz and Alan Richter have, in fact, recently used this technique to improve the performance of an isolated-word recognition system [Poritz 86].

There are clearly all kinds of variations to this scheme. Poritz and Richter, for example,



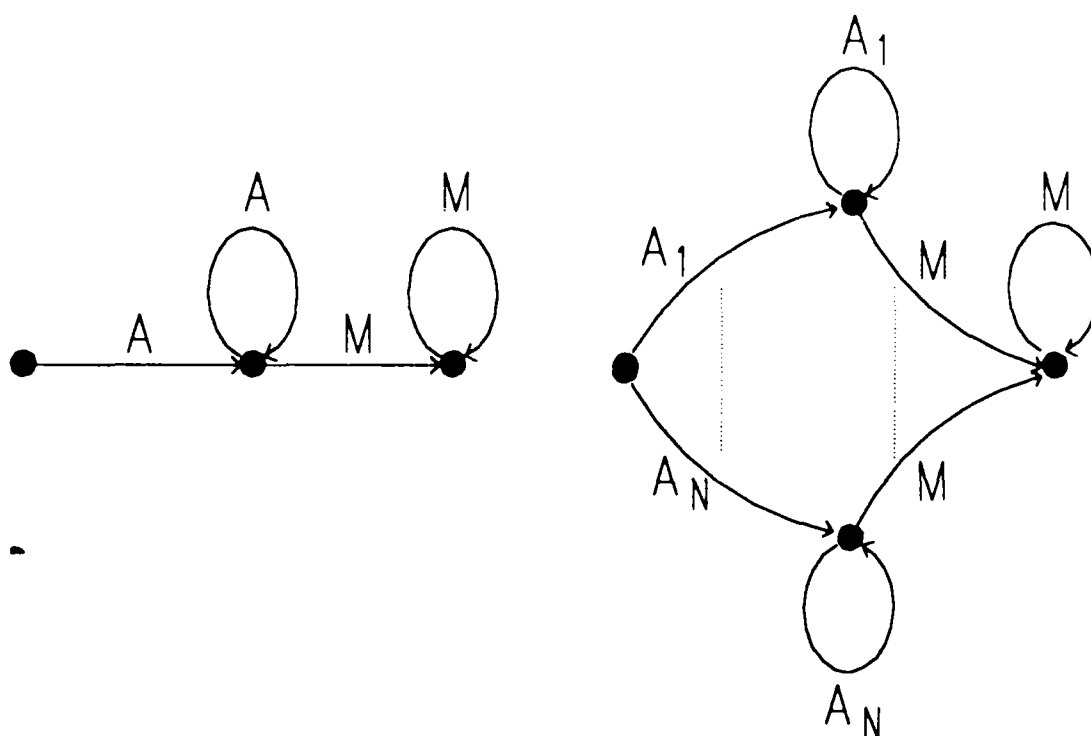


Figure 7.1: Time-Derivative Modeling by Machine Alteration

found that they maximized their performance by adjoining the current acoustic parameter vectors with the ones which had been produced eight centiseconds earlier. One might also consider adjoining three or more sets of parameters together.

The only problem with this technique is that it increases the dimension of the parameter vectors that are to be modeled by the acoustic models, and, as we discussed in the previous section, this increases the variances of the probability density estimates that are obtained from these models. Fortunately, however, this is a problem for which we have an antidote; we can extract from these adjoined vectors of parameters a small number of information-bearing components by projecting them onto principal discriminants. If it turns out that the time-derivative information is less important than the information in the original spectrum then, hopefully, this information will be discarded. If, on the other hand, it is more important than some of the original spectral information, then it will be retained and some of the original information will be discarded. The parameter-reduction method described in the previous section, therefore, allows us to adjoin acoustic-parameter vectors freely without having to worry about overdoing it.

## 8. Conditional Models

### 8.1 The Correlation Problem

In Chapter 3, we described one of the basic assumptions of hidden Markov models, the output-independence assumption. It states that the probability that an acoustic observation,  $y_t$ , will occur at time,  $t$ , depends only on the output distribution on the transition taken at time  $t$ . In this chapter, we will examine this assumption and certain alternative assumptions.

Consider a system which uses one output distribution for each English phone or allophone. Assuming an alphabet of, say, 50 phones, the system would have 50 output distributions. From the point of view of such a system, speech would be generated by using the language model and the word models to generate a sequence of phones, and then by generating an acoustic observation sequence from the phone sequence. The transition probabilities in the phone models would be used to determine the durations of each instance of a phone. The acoustic observation at a given time would be generated randomly from the output distribution for the phone that was being produced at that time. The fact that the observation vectors produced each centisecond by the signal processor are correlated in time would be modeled by the transition probabilities from one phone to another, and more importantly by the transition probabilities which determine the durations of the different phones. The problem with such a system is that for a number of reasons, there is normally a significantly larger amount of correlation than can be modeled in this way.

The tongue and the other articulators have inertia; they do not move instantly from one position to another. As a result, the sound which is produced at the beginning of a phone depends on the previous phone, the sound which is produced at the end of a phone depends on the following phone. Whole phones can be nasalised if nearby phones are nasals. Different speakers speak differently, and the quality of a speaker's rendition of a particular sound effects the whole rendition. The quality of an individual speaker's voice can also change from rendition to rendition. If a speaker is speaking loudly, he will stay speaking loudly for some time, he doesn't alter the volume of his voice randomly on a centisecond by centisecond basis as the above model assumes. The environment varies slowly over time. If a fan is on in the background, it will be on for a while, and will not be turned on and off randomly each centisecond. There are all kinds of slowly varying processes which can effect

the acoustic features produced by the signal processing component of a speech recognizer.

Ideally, one would like to use some sort of sophisticated signal processing which produces features that are independent of long-term acoustic effects. To some extent this is what has been attempted in the ear-model component of the IBM signal processing. At the moment, however, no one knows how to do this well, and until someone does, the acoustic models used in a speech recognizer will be faced with the problem of modeling the influences of long-term acoustic phenomena.

One might try to alleviate this correlation problem by introducing more states into the hidden Markov models. So, for example, instead of using just one set of states and one output distribution to model the *i* phone, one might consider using models in which there are different output distributions and sets of states, for loud *i*'s, soft *i*'s, *i*'s with fans in the background, nasalized *i*'s, nasalized *i*'s with fans in the background, and so on. The problem with this approach is that it would require a tremendous number of states and output distributions. This, in turn, would create a need for an exorbitant amount of training data and computational horsepower.

The only way to avoid using separate sub-phones which represent many different variations in the pronunciation of a phone, is to alter the output-independence assumption. If we want to model long-term phenomena which cause the output of the signal processing component during one centisecond to resemble the output during the previous centisecond, without explicitly modeling each possible combination of such phenomena, then we must do away with the assumption that given the output distribution at time  $t$ , the acoustic observation at time  $t$  is independent of that at time  $t - 1$ . The replacements for the output-independence assumption that one might consider depend on whether the system is based on a family of discrete or of continuous hidden Markov models, and we shall therefore consider the two cases separately.

## 8.2 The Continuous Case

We would like to alter the output-independence assumption to capture the fact that with a small number of output distributions, the way that  $y_{t-1}$  differs from the mean of the output distribution from which it is generated influences the way that  $y_t$  differs from the mean of the output distribution from which it is generated. One could do this directly by conditioning the probability of generating  $y_t$  on the transition at time  $t$ , the transition at time  $t - 1$ , and on  $y_{t-1}$ . The output-independence assumption in Chapter 3 would then

be replaced by

$$\begin{aligned} \Pr(Y_t = y_t \mid Y_1^{t-1} = y_1^{t-1}, X_1^T = x_1^T) \\ = \Pr(Y_t = y_t \mid X_{t-1} = x_{t-1}, X_t = x_t, X_{t+1} = x_{t+1}, Y_{t-1} = y_{t-1}). \end{aligned} \quad (8.1)$$

To incorporate such an assumption into a hidden Markov model, it would, in effect, be necessary to square the number of output distributions, and, in addition, to increase the number of parameters in each output distribution. If we assume no tied output distributions and no zeroed-out transition probabilities the computation in the forward-backward algorithm, as applied to models incorporating the output-independence assumption, increases linearly with the number of transitions in the model. If we, instead, use a model incorporating (8.1), then the computation will increase as the square of the number of transitions.

An alternative to (8.1) is to condition only on the transition taken at time  $t$  and on the output produced at time  $t-1$ . This could be done by replacing the output-independence assumption with

$$\begin{aligned} \Pr(Y_t = y_t \mid Y_1^{t-1} = y_1^{t-1}, X_1^T = x_1^T) \\ = \Pr(Y_t = y_t \mid X_t = x_t, X_{t+1} = x_{t+1}, Y_{t-1} = y_{t-1}). \end{aligned} \quad (8.2)$$

From a computational view this assumption is much more attractive because it does not involve increasing the number of output distributions, only the number of parameters in each individual output distribution. From a modeling point of view, however, this assumption is much less attractive because if we don't know the distribution from which  $y_{t-1}$  was generated, we don't know how  $y_{t-1}$  differs from its mean.

In order to get a better feeling for this assumption, let us examine the form of an output distribution that might be used in a model based on (8.2). Assume that the  $n$ -dimensional random variables  $y_{t-1}$  and  $y_t$  are jointly Gaussian. It is shown in Appendix 13.5 that  $d$ , the distribution of  $y_t$  given  $y_{t-1}$ , is of the form

$$\begin{aligned} \Pr(Y_t = y_t \mid Y_{t-1} = y_{t-1}) \\ = \frac{|W|^{1/2}}{(2\pi)^{n/2}} e^{-\frac{1}{2}(y_t - (\mu_2 + C(y_{t-1} - \mu_1)))' W (y_t - (\mu_2 + C(y_{t-1} - \mu_1)))}, \end{aligned} \quad (8.3)$$

which is just an  $n$ -dimensional Gaussian with mean  $\mu_2 + C(y_{t-1} - \mu_1)$  and covariance matrix  $W^{-1}$ . Note that there are less than three times as many parameters in this distribution as there are in an ordinary  $n$ -dimensional Gaussian. The forward-backward reestimation formulas for  $\mu_1, \mu_2, W$ , and  $C$  are in also in Appendix 13.5, as are the derivatives necessary for MMIE.

The observation vector  $y_t$  is assumed to have been sampled from a Gaussian,  $d$ , with a mean that depends on how  $y_{t-1}$  deviates from  $\mu_1$ . But because the model does not remember which output distribution was used at time  $t-1$ ,  $\mu_1$  is not necessarily the mean of the distribution from which  $y_1$  was actually sampled. Rather  $\mu_1$  is an average of all the samples which precede by one unit of time a sample generated from  $d$ . Let  $i$  be a state which serves as the origin for a transition which has output distribution  $d$ . Let  $j$  be another such state. If it was true that the distributions for all the samples which contributed to  $\mu_1$  had the same mean, then the word models would be such that all the means of the output distributions on all transitions entering both  $i$  and  $j$  would be the same. Hidden Markov models which have such a property either have very limited tying of output distributions, or have very uninteresting graph structures. In either case such models are not very useful for large vocabulary speech recognition. Equation (8.2) will, therefore, be inaccurate when applied to large vocabulary speech recognition.

Nevertheless, it is bound to be more accurate than the original output-independence assumption. Normally, in models used for speech recognition, acoustic observation samples tend to be generated from the same output distribution for a number of consecutive time frames. In such models the  $\mu_1$ 's will tend to be averages of samples generated from the  $d$ 's, and most of the time they will be close to what they should be. So, although we know that (8.2) is bound to be inaccurate, it may not be that much less accurate than (8.1), and certainly is much more favorable from the point of view of estimation error and computation. It is an empirical question whether or not the increased potential accuracy of a model based on (8.2) can be realized in a system which must estimate parameters from limited training data.

### 8.3 The Discrete Case

There is strong evidence, at least in the IBM system which uses discrete hidden Markov models, that the output-independence assumption is inaccurate. This evidence comes from examining two different types of models. The first is the standard IBM family of models. Using the method described in the Chapter 3, we can compute the probability of generating a sequence of acoustic labels with this family given the word sequence in the corresponding script. This is done by concatenating word models together according to the script to form one large model, and then using the forward-pass of the forward-backward algorithm to compute the probability that this model will generate the acoustic label sequence. Now

consider a second acoustic model which simply models a sequence of acoustic labels as a first-order Markov process. We can obtain maximum likelihood estimates for this model simply by counting occurrences of pairs of acoustic labels in a training sample. We can also use this model to compute the probability of a speech sample,  $y_1^T$ ,

$$\Pr(Y_1^T = y_1^T) = \Pr(Y_1 = y_1) \prod_{i=2}^T \Pr(Y_i = y_i | Y_{i-1} = y_{i-1}). \quad (8.4)$$

Unlike the family of standard models, this model has no concept of word or phone and is therefore useless for speech recognition. Yet it turns out that the probability of a sample of speech as computed with this second model is generally significantly greater than the conditional probability of the same sample given the sample script as computed with the family of standard models. Clearly, then, the output-independence assumption as used in the IBM system is not accurate.

If the output-independence assumption is replaced by (8.1) in a discrete hidden Markov model, in essence, there will have to be a separate output distribution for each original pair of output distributions. Furthermore, each such output distribution will have to provide a probability for every pair of acoustic labels. This amounts to effectively squaring the number of output distributions, and squaring the number of parameters in each distribution. The IBM system which contains approximately 200 output distributions, each with 200 parameters has approximately 40,000 free parameters. If (8.1) were put into the IBM system, the system would then have approximately 1.6 billion free parameters, which is clearly too many. So again, we must consider alternatives.

As in the continuous case, we can drop the conditioning on the previous distribution and only condition on the previous acoustic label. In this case, the only change is to square the number of parameters in each output distribution. There would then be approximately 8 million free parameters in the IBM system. This is still too many.

Somehow, we must tie certain probabilities together so as to reduce the number of parameters in the models. One method would be to tie whole conditional distributions together. So, for example, in output distribution  $d$ , we might insist that for acoustic labels  $i$  and  $j$

$$\Pr(Y_t = y_t | Y_{t-1} = i) = \Pr(Y_t = y_t | Y_{t-1} = j). \quad (8.5)$$

More generally, for each distribution  $d$ , we could define a function which maps acoustic label  $i$  to a class of acoustic labels  $C_d(i)$ . We could then insist that

$$\Pr(Y_t = y_t | Y_{t-1} = y_{t-1}) = \Pr(Y_t = y_t | Y_{t-1} \in C_d(y_{t-1})). \quad (8.6)$$

At one extreme, if we have only one class per distribution, then we will discard all the information that an acoustic label at time  $t - 1$  can provide about an acoustic label at time  $t$ , except that which is retained in the state of the Markov chain. At the other extreme, if we have as many classes as there are acoustic labels, we will be stuck with an enormous number of model parameters that can not be reliably estimated, making the information we do have inaccurate. If we have an average of  $n$  classes per distribution, then the number of parameters in the models would be increased by a factor of  $n$ . We can thus choose  $n$  to give us whatever position we desire on the trade-off curve between a model which is inherently inaccurate and one which has so many free parameters that it will be inaccurate in practice. We may also want to have different numbers of classes for different distributions.

Let us now address the problem of determining the classes of acoustic labels. Suppose we are given a sample of training data: a script of the words spoken, and a sequence of acoustic labels which represents the system's representation of the speech in the sample. To make our model as accurate as possible, we would, ideally, like to define our classes in such a way that we maximize the average amount of additional information that the class of one acoustic label provides about the next acoustic label. For simplicity, assume that we know which centisecond-long frames in the sequence were generated from which output distributions. We can make a reasonable assignment of frames to output distributions by obtaining a Viterbi alignment between the frames and the transitions in the model corresponding to the training script, as described in Section 3.1. Given such an assignment, our problem then reduces to determining for each output distribution a set of classes of acoustic labels which maximizes the average information between the class of a label and the next label, where this average is taken over all frames assigned to that output distribution.

We can now produce classes of previous acoustic labels for a given distribution,  $D$ , in a greedy fashion. We begin by assigning each acoustic label to its own class. If we have an alphabet of  $A$  acoustic labels, then, initially, we will have  $A$  classes. For each class we can compute from our training sample the conditional distribution of current labels given that class. From these distributions, we can compute the average mutual information, between the current label and the class of a previous label. Now, suppose we merge two classes. On average, we certainly cannot gain any information by forgetting to which of these two classes an acoustic label belongs. Furthermore, if the conditional distributions corresponding to these two classes differ, then we will necessarily lose some information. So, in general, if we decrease the number of classes by merging two classes together, we will also decrease the average mutual information between the current label and the class of the previous label.

For our purposes, it will, therefore, be best to merge the two classes which minimize this loss of information. Each time we merge two classes together, we decrease the number of classes for a given distribution by one. We can produce  $n$  classes for distribution  $D$  by performing  $A - n$  such merges. This method of creating classes was originally formulated by Robert L. Mercer. Together with John Lucassen, he also developed a fast implementation of his method which is described in [Lucassen 82].

For some distributions, we will lose more information by reducing to a fixed number of classes than for others. It seems reasonable that for these distributions it would be better, in fact, to have more classes. An alternative to creating a fixed number of classes for each distribution, would be to reduce the mutual information by a fixed percentage for each distribution. Another alternative would be to merge classes for all distributions at once by choosing the distribution and the pair of classes to merge according to which results in the least loss of information across all distributions, and all pairs of classes.

Peter V. deSouza and I tested a conditional model of the type described above by modifying the detailed match in a 2000-word-vocabulary, speaker-dependent, isolated-word experiment. Both the training and the test samples consisted of 207 sentences, each containing 10 random words. A language model with a uniform distribution was used to decode the test sentences. The word models used in this experiment contained a collection of 200 output distributions, and an alphabet of 200 acoustic labels was used by the signal processing component of our system. For each distribution, we used the program developed by Mercer and Lucassen to merge classes of acoustic labels until we had fewer than 6 classes, and had lost at least half the average mutual information between an acoustic label and the class of the previous acoustic label. This resulted in an average of 2.5 classes per output distribution, and increased the number of free parameters in the collection of output distributions from approximately 40,000 to 100,000. The forward-backward algorithm was run for 3 iterations to train the statistics in the resultant conditional hidden Markov models. A fourth smoothing iteration was then performed using the method of interpolated estimation described in [Jelinek 80]. In comparing this conditional model to the original unconditional model, the probability of generating the training data increased from an average of .098 per frame to an average of .117 per frame. This is no surprise in itself since the conditional model has 2.5 times as many parameters as the original model, but it does indicate that the conditional model is potentially more accurate. On the 2070 words of test data the number of errors made by the detailed match decreased from 46 to 34, which strongly indicates that the conditional model is, in fact, a more accurate model.



#### 8.4 Run-Length Adjustment

In this section, we will consider another very simple attack on the correlation problem in the discrete case. One major effect of the correlation problem is that steady-state regions of speech tend to receive too much emphasis. Consider vowels, for example. Although there are a variety of ways that a given vowel might be produced, a given production often contains a relatively long period of steady-state sound. If, during the production of a vowel, a few consecutive centiseconds of a particular acoustic label are observed, then it can often be predicted with fairly high reliability that during the next centisecond that particular label will again be observed. If this vowel is modeled by a single output distribution then because this output distribution must model all the different ways that the vowel can be pronounced, it will tend to assign too small a probability to the label produced during this centisecond. If we think of the cost of generating a word as the negative of the logarithm of the probability of generating the word, then a large part of the cost of generating words with hidden Markov models which are based on the output-independence assumption will be associated with the production of vowels; a much larger cost than should be associated with them. This observation is corroborated by examinations of the errors which are made by the IBM system. There is a strong feeling among the people working on this system that it puts too much emphasis on vowels. It tends to ignore gross differences in consonants but rarely misses vowels. No doubt vowels are simpler acoustic events than consonants, and can therefore more easily be modeled and discriminated from one another. But the point here is that we would have predicted this anyway from an understanding of the correlation problem.

If we are right about the effect of the correlation problem in steady-state regions of speech, then we would expect to observe longer runs of acoustic labels than would have been predicted by our models. This is something that can be tested empirically. If we take a sample of speech and a corresponding script, we can use the forward-backward algorithm to estimate the parameters of a hidden Markov model corresponding to the training script. We can then randomly generate an acoustic-label sequence using the transition and output probabilities in the trained model. For each acoustic label,  $i$ , we can compute the average run length of strings of consecutive  $i$ 's in this random sequence. We can also determine the average run length of strings of consecutive  $i$ 's in the training sample. Table 8.1 shows averages of ratios of sample run lengths to IBM-model run lengths for one speaker. The sample for this experiment was recorded in an environment which was relatively noise-free.

Acoustic labels were associated with the phones that they tended to be produced from most often, as determined by a Viterbi alignment. They were then grouped into classes according to the identity of the phones. The maximum ratio observed was 4.2 for a label which was most often associated with the vowel in *put*. The minimum ratio observed was .94 for a label which was most often associated with background noise. Notice that, as predicted, the ratios tend to be high in vowels and other steady-state sounds, such as semivowels, and lower in shorter sounds, such as fricatives and stops. As we should expect in a sample which is free of long-term noise the ratios for the noise patterns are all very close to unity. Overall, the average ratio is 1.6 which is large enough to serve as strong evidence that our models do, indeed, suffer from a correlation problem.

<i>Class</i>	<i>Mean</i>	<i>Standard Deviation</i>
Vowels	1.80	.439
Semivowels	1.81	.358
Nasals	1.69	.402
Fricatives	1.55	.323
Stops	1.31	.300
Noise	1.00	.066

Table 8.1: Average Ratios of Observed to Predicted Run-Length

In order to understand a simple adjustment that can be made for this effect, consider the following hypothetical example. Suppose we have a 3-state, 2-transition machine that generates sequences of length 2. The first transition,  $1 \rightarrow 2$ , generates the first element in the sequence, and the second transition,  $2 \rightarrow 3$ , generates the second element. Suppose that for the data this machine is modeling, every time label  $\ell$  appears as the first element, it also appears as the second element. After training the machine, the probability,  $p$ , of each transition generating label  $\ell$  will be equal to the probability of the sequence being a sequence of two consecutive  $\ell$ 's. Suppose further that such a sequence of  $\ell$ 's is relatively rare so that  $p$  is relatively small. The observed average run length of sequences of  $\ell$ 's will be twice as great as that predicted by the model. Furthermore, the machine will generate a sequence of  $\ell$ 's with probability  $p^2$ , even though the true probability of such a sequence is  $p$ . If the sequences of  $\ell$ 's had been always of length  $n$ , and we were using a similar model with  $n$  transitions, then the ratio of observed to predicted run lengths would be  $n$ , and the probability of generating a sequence of  $\ell$ 's with the model would be the  $n$ th power of

what it should be. This suggests that we can perform a rough correction for the correlation problem by raising the output probabilities of a label  $l$  to the  $R(l)$ th power, where  $R(l)$  is the ratio of predicted to observed average run lengths of sequences of  $l$ 's. The argument given above is very heuristic and changes if we, for example, change the machine shape or the distribution of sequences of consecutive  $l$ 's. The resulting correction is therefore very crude, and the probabilities that the resultant models assign to acoustic label sequences will, no doubt, be inaccurate. However, the probabilities assigned to these sequences by the uncorrected models are certainly way off, and some improvement, albeit crude, would be better than none. Furthermore, it is a simple change, and only increases the number of free parameters by the size of the acoustic alphabet, which in the IBM case amounts to an increase by less than half a percent.

Following the argument above, if we do not take any probabilities to any powers, on average the probability of generating an acoustic sequence, given a word sequence, will be some power of what it should be. When decoding, we multiply the acoustic match probability,  $\Pr(Y_1^T = y_1^T \mid W_1^T = w_1^T)$  by the language model probability,  $\Pr(W_1^T = w_1^T)$ , to obtain the joint probability,  $\Pr(Y_1^T = y_1^T, W_1^T = w_1^T)$  as described in Section 2.1. But if  $\Pr(Y_1^T = y_1^T \mid W_1^T = w_1^T)$  is some power of what it should be, the language model component will be given short shrift. Empirically, it has been found in the IBM system, that better results are obtained when the following approximation is used [Bahl 80]

$$\Pr(Y_1^T = y_1^T, W_1^T = w_1^T) \approx \Pr(Y_1^T = y_1^T \mid W_1^T = w_1^T)^{.25} \Pr(W_1^T = w_1^T). \quad (8.7)$$

The exponent of .25, which is referred to as the *language model match factor*, further indicates that the correlation problem is, in fact, a serious one in the IBM system.

The run-length correction described above can be thought of as adjusting the language model match factor according to the amount of unaccounted for correlation in  $y_1^T$ , according to the acoustic labels in  $y_1^T$ . It can be implemented in the following steps. First, estimate the parameters in a family of hidden Markov models with the forward-backward algorithm in the normal way. Next, use the resultant models and the training sample to compute ratios of predicted to observed average run lengths for each acoustic label. Raise all output probabilities for an acoustic label to the run length ratio for that label. Finally, use the adjusted output probabilities in the normal way to recognize speech. It also makes sense to retrain the transition probabilities in the model after the output probabilities have been adjusted. In practice, however, it was found that this retraining has little effect.

It may seem troublesome that after adjustment the *probabilities* of generating different acoustic labels in an output distribution no longer sum to unity, and as a result, the sum over

all label sequences of the *probability* of generating a sequence also no longer sums to unity. The adjusted model, then, is no longer a true probability model. Although it certainly unacceptable in the long run, this problem is not as serious as it seems. The reason the sum is no longer unity is that the probabilities of all sequences have been increased. The probabilities of the speech-like sequences have been increased as they should be. However, the probabilities of the sequences that are not speech-like, those containing shorter runs, have also been increased, whereas they should have been decreased. In practice, we only care about the speech-like sequences, and the fact that our models incorrectly model sequences which are not representative of actual speech is of secondary concern.

An initial experiment with run-length adjusted statistics was performed on the random-word task that is described in the previous section. In this experiment the number of detailed-match errors was decreased from 46 to 32. This is even a slightly larger reduction in the error rate than was obtained with the conditional models described in the last section. Although the model is less sophisticated here, it may have performed as well as the model in the previous section because it has significantly fewer free parameters.

Next, an experiment on a 20,000-word-vocabulary, speaker-dependent, isolated-word, natural-language task with a trigram language model was performed. Four speakers were tested, each reading training and test scripts of 100 sentences. In this experiment, which used an informative language model, the language model match factor was increased from .25 to .4 to account for the fact that much of its job was already being done by the output probability adjustments which averaged .625. Originally, before adjustment, there were 44, 60, 84, and 45 errors for the four speakers. After adjustment according to the procedure described above the numbers of errors decreased to 42, 50, 79, and 38, respectively. The total number of errors across the four speakers dropped from 233 to 209, which amounts to reduction in the error rate of the system from 3.43 percent to 3.08 percent.

## 9. An LPC-Based Waveform Model

### 9.1 The Framing Problem

Throughout this thesis we have approached the problem of acoustic modeling in speech recognition as that of extracting as much information as possible from a speech sample about the corresponding word sequence. In Chapter 2 it was shown that this information is bounded from above by the mutual information between the input speech signal and the corresponding word sequence. The fact that human performance is currently significantly better than machine performance is strong evidence that we are not currently achieving this upper bound. We have seen that there are two possible reasons for this: either our acoustic models are inaccurate, or we are discarding information in the signal-processing component of the system.

Ideally, the signal-processing component of a speech recognizer will convert the acoustic signal it receives into a form which can be modeled accurately and which contains as much information about the corresponding word sequence as does the original waveform. There is evidence, however, that in the IBM system the signal-processing component is discarding information. In one experiment it was observed that nearly half of the recognition errors involved nasalized *d* endings in word pairs such as *plan* and *planned*, *an* and *and*, and *contain* and *contained*. The *d* bursts, which were clearly present in the original waveforms, were difficult, if not impossible, to see in the spectra produced every centisecond and could not be detected in the acoustic labels. In another experiment involving the speaker-independent recognition of the four letters *b*, *d*, *e* and *v* researchers at IBM were having trouble getting their system to approach human performance which had been measured at 94 percent. In an effort to determine where their problems were located, they synthesized speech waveforms from the spectra computed every centisecond in the signal-processing component of the recognizer. Human performance on these synthesised waveforms dropped to 75 percent. Furthermore, the dominant human error was the same as the dominant error made by the computer, *b* being misrecognised as *v*.

One explanation for these results is the random placement of the signal-processing window with respect to the *b* and *d* bursts. Recall that the signal-processing component of the IBM system begins by sampling the analog waveform at 20,000 Hertz, and then by computing a series of short-term spectral estimates. Each spectral estimate is computed

over a window of 512 sample points. This window is moved forward by 200-point increments so that a fresh spectral estimate is computed every centisecond. Because acoustic events are in no way synchronized with the system clock, the placement of this window, or frame, with respect to an acoustic event such as a *b* burst is purely random. Furthermore, the effective size of the Hamming window does not vary with respect to the acoustic events that are being windowed. The contributions to the spectrum of short-duration events such as *b* and *d* bursts are often averaged together with the contributions of surrounding events because the window is so long that it does not isolate the short-duration events in time. The fixed size of this window cannot simply be decreased, however, without losing spectral resolution in steady-state events such as vowels.

Ideally, we would like to window acoustic events with windows that are placed and sized according to the events themselves. Although it should be possible to obtain an improvement by attempting to window the speech waveform in an event-synchronous fashion, such an approach suffers from a common problem in pattern recognition: in order to detect an event, one would like to know what event it is that is being detected, but in order to decide what the event is, one would like to know where it is. If it were possible to decide with certainty in the signal-processing module that an event such as a *d* burst had occurred, then we would barely need the acoustic-modeling and language-modeling components which constitute the rest of the recognizer. If, on the other hand, this module can not be sure whether or not it has detected such an event, then it risks discarding information by assuming that it has.

The essence of this framing problem is that in order to extract just the right information from the acoustic signal the signal-processing component must make certain hard decisions without having at its disposal all the information relevant to such decisions. The probability that an acoustic event has occurred at some point in the acoustic signal depends not only on that particular section of the signal, but also on the surrounding acoustic and linguistic events. In fact, it depends on just those events that the rest of the system is designed to model. This suggests that we may find an answer to this problem by embedding the signal processing into the statistical models used by the rest of the system.

If we model the digitized acoustic waveform directly, we know that no relevant information will be lost in the signal-processing component of the system because there will be no separate signal-processing component. There will be no framing, and hence no framing problem. If the underlying Markov model makes a state transition each waveform sample, then an event such as a *d* burst will be detected just when the model transits into a *d*-burst

state, and this will be influenced by all the power of the rest of the recognizer, including, for example, the language model. Events will be detected and recognized simultaneously. While these potential benefits are no doubt very large, in order to realize such a model we have to face up to a nasty problem: modeling the waveform directly.

## 9.2 Linear Predictive Models

By far and away the most popular class of statistical distributions which is capable of directly modeling a speech waveform in a sample-by-sample fashion is the class of distributions used in *linear predictive coding* (LPC) [Markel 76] [Makhoul 75]. In an  $n$ th-order linear predictive model, a waveform sample is predicted as a linear function of the previous  $n$  samples:

$$y_t = \sum_{j=1}^n a_j y_{t-j} + \epsilon_t. \quad (9.1)$$

The error term  $\epsilon_t$  is assumed to have a Gaussian distribution. The coefficients  $a_1, \dots, a_n$  are referred to as *linear prediction coefficients*.

Linear predictive models have been successfully used for speech compression, vocoding, and speech synthesis for over a decade. In 1973 Itakura obtained impressive results using linear predictive models in speech recognition [Itakura 75]. In his system, and in the many similar ones which have followed, it is assumed that short intervals of speech are generated from stationary linear predictive models. Each centisecond, or so, a set of linear predictive coefficients is estimated from a short window, or frame, of speech. Frames of speech are then compared to sound templates with what has come to be known as the *Itakura metric*, which is an approximation, see [Thomson 85], of the ratio of the probability that the speech in a frame was generated from an LPC model based on a predetermined set of coefficients to the probability that the speech was generated from an LPC model based on the coefficients estimated from the speech itself is then computed. Notice that although these systems use LPC models, they are not based on waveform models in the sense that we have been discussing. In particular, they still window the acoustic signal in a manner which is independent of the recognition of acoustic events.

Nevertheless, the basic LPC model does model a speech waveform sample by sample, and in this respect, it is just what we need. A hidden-Markov LPC-based waveform model could be constructed in the following manner. Each sample of the waveform, the underlying Markov chain would make a state transition, and given that a sample is generated

from a particular transition, the probability density at that sample would be determined from a linear predictive output distribution which is associated with that transition. The parameters for each output distribution would be a set of linear prediction coefficients and a variance for the error term. Consider an output distribution with coefficients  $a_1, \dots, a_n$  and error-term variance of  $\sigma^2$ . Using this output distribution, the error term at a waveform sample  $y_t$  would be

$$\epsilon = y_t - \sum_{j=1}^n a_j y_{t-j}. \quad (9.2)$$

The probability density at  $y_t$  would be

$$N(\epsilon : 0, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\epsilon^2}{2\sigma^2}}. \quad (9.3)$$

The transition probabilities, the prediction coefficients, and the error-term variances can be estimated either with the MLE or MMIE, in the usual way. Forward-backward reestimation formulas for such a linear predictive model, as well as the derivatives for MMIE hill-climbing can be found in Appendix 13.6.

The LPC model describes the output of a series of  $n$  lossless acoustic tubes driven by Gaussian noise. While this model is taken quite seriously as a model of speech, it has various flaws. One particularly serious one is that it does not model the periodic driving function of the vocal chords during voiced speech. Following a suggestion of John Makhoul's [Makhoul 86], we can attempt to overcome this problem by using a pitch-period detector and placing a few coefficients around waveform samples one pitch period,  $p_t$  samples, away. The model then becomes

$$y_t = \sum_{j=1}^n a_j y_{t-j} + \sum_{j=-m}^m b_j y_{(t-p_t+j)} + \epsilon_t. \quad (9.4)$$

The effects of this pitch-period modeling can be seen by examining the three figures on the next page. In these figures the x-axis is in samples, where the sample rate is 20,000 Hertz. The top figure shows the first few centiseconds of the acoustic waveform for a rendition of the letter *v*. The next two figures display the log of the probability of producing each sample from the state that a hidden Markov model was most probably in at the time of that sample. The two LPC-based hidden Markov waveform models in this example were trained with the forward-backward algorithm. Figure 9.2 shows these log probabilities for an LPC model corresponding to equation (9.1) with  $n = 30$ . Figure 9.3 shows these log probabilities for an LPC model corresponding to equation (9.4) with  $n = 30$  and  $m = 2$ . The pitch periods



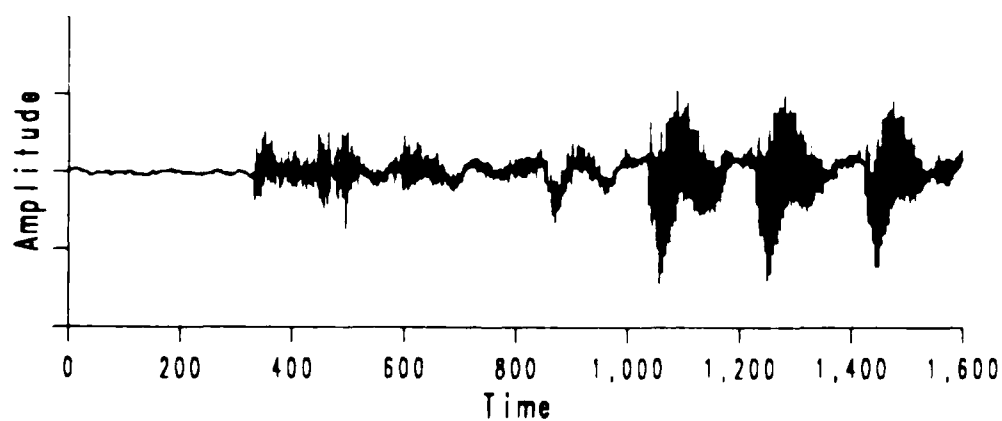
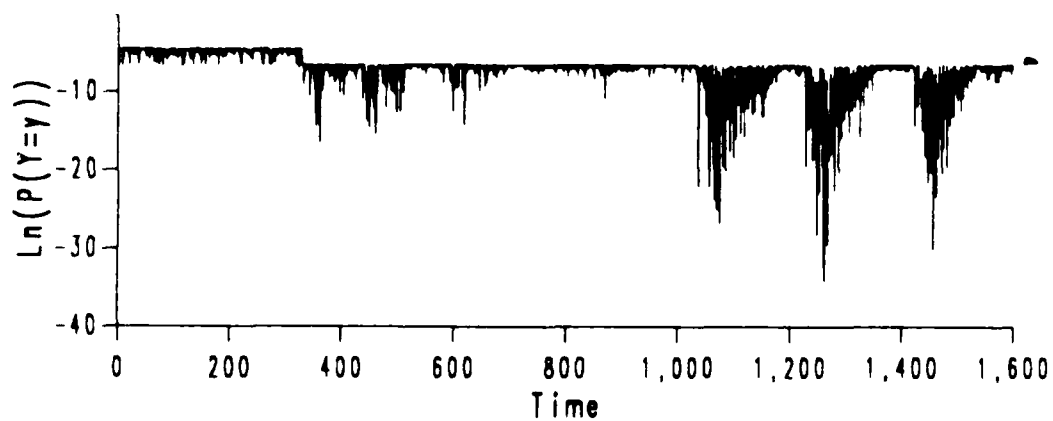
Figure 9.1: Initial Section of  $v$  Waveform

Figure 9.2: Scores from Basic LPC Model

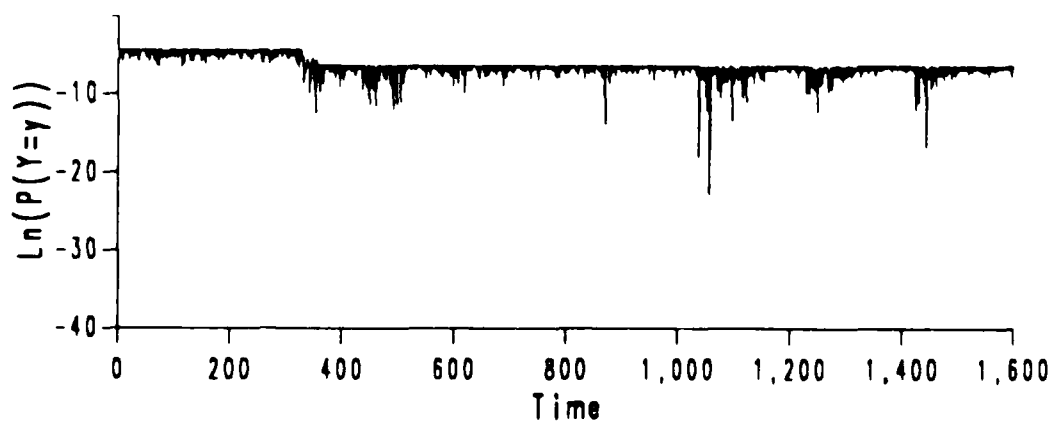


Figure 9.3: Scores from LPC Model with Pitch Tracker

for this model were computed with a standard autocorrelation pitch tracker. Both models change from a state which models background noise to a state which models the *v* fricative at about the 340 sample point. Since the noise distributions have tighter variances than the other distributions in these models, the upper bound on the probabilities drops at this point, as is clear in the figures. In Figure 9.2 the log probabilities are extremely low around the high amplitude speech in the beginning of the vowel. Each new pitch period comes as a complete shock to this model. In Figure 9.3, however, this effect is greatly diminished. There are still a few samples which are modeled very badly, but not nearly so many as in the preceding case.

There is another problem with LPC waveform models which can not be so easily remedied. In the current IBM system, a state transition occurs every centisecond, and output probabilities are simply looked up. In the waveform model described here, 20,000 state transitions occur each second, and the evaluation of an output probability density involves a, say, 40-point dot product, in addition to the evaluation of a Gaussian density. This is roughly 10,000 times as much computation as is involved in the current system. This is a serious problem. It is not simply that it will be difficult to create a real-time system based on a waveform model, but it will be difficult to run experiments at all.

Nonetheless, the idea of modeling the speech waveform directly has so many potential advantages that a very small and very preliminary experiment was conducted. The task was the speaker-independent recognition of the four letters *b*, *d*, *e* and *v*. Although this task only has a 4-word vocabulary, these four words are particularly difficult to discriminate from one another, as is evidenced by the 80 percent performance of the current IBM system on this task. In order to reduce the amount of computation required, salient sections were clipped out of the speech waveforms. Using a Viterbi alignment produced by the standard IBM recogniser, the transition into the vowel was located for each sample word. Fifteen consecutive centiseconds of speech were extracted from each word, ten before the vowel transition, and five after. These 15-centisecond-long waveform segments became the samples for this waveform-model experiment. Since the speech signal was sampled at 20,000 Hertz, each such segment consisted of 3000 points.

In a waveform model, acoustic events are modeled on a much smaller scale than in the standard IBM system. It is, therefore, not clear at all what the shapes of the finite-state machines for the individual words should be. We would like them to have many states in order to model the detail in transitional events. But, as we have seen time and time again, the quality of our density estimates diminishes with the number of parameters in

the system. Furthermore, the computation increases with the number of transitions in the machines, and in this experiment we have to be very careful about computational increases. Each phone in a set of phonetic word models was modeled with a four-distribution machine as shown in Figure 9.4. The phonetic spellings for the four words are shown in Table 9.2. Note that it is not necessary to model the vowel trail-offs or end-of-word silences because the final sections of the words have been clipped out.

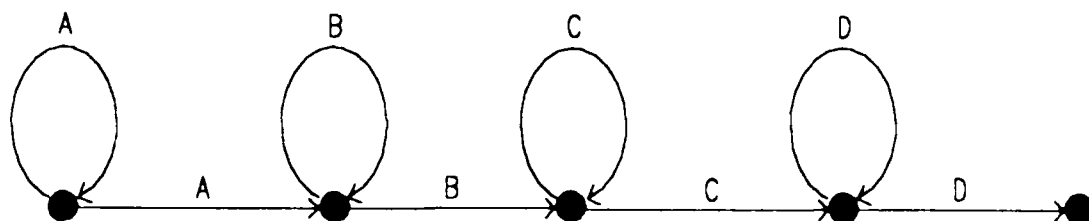


Figure 9.4: Waveform Phone Machine

Word    Phonetic Representation

<i>b</i>	Noise Onset <sub>1</sub> B E
<i>d</i>	Noise Onset <sub>1</sub> D E
<i>e</i>	Noise Onset <sub>2</sub> E
<i>v</i>	Noise Onset <sub>3</sub> V E

Table 9.1: Phonetic Spellings for Waveform Experiment

Each word was uttered twice by 100 different speakers. Once for the training data, and once for the test data. Words which were clearly misspoken were discarded, leaving 372 words of training data and 396 words of test data. The basic LPC model of equation (9.1) with  $n = 30$  was tested, as well as a version of the model described in equation (9.4), with  $n = 30$  and  $m = 2$ . The pitch periods, the  $p_t$ 's, were determined by an autocorrelation-based pitch-period detector. Parameters were estimated with the forward-backward algorithm. MMIE was implemented, but was found to involve a prohibitive amount of computation for this preliminary experiment. The performance on test data of the basic model was 64.9 percent, and the performance of the pitch-period enhanced model was 65.4 percent.

There are a number of potential explanations for these disappointing results. First, as we can see from the log probabilities in Figures 9.2 and 9.3, error terms do not seem to

have Gaussian distributions. In particular, there are many more outliers even in the model which attempts to model the periodicity of the source. This is just the situation for which Richter models are designed, and in any future attempt, it would be reasonable to try to model the error term as a Richter mixture of Gaussians. A second potential source of poor performance may be the word models. When attempting to discriminate between *b*, *d*, *e* and *v* with models in which the vowel distributions are tied together, all that matters are the consonants and the transitions into the vowels. But it is certainly far from clear how many states and output distributions are needed to model these rapidly varying acoustic events on a sample-by-sample basis. Unfortunately, developing accurate word models is to a large extent a matter of trial and error, and at the present time we do not have the computational resources to perform the necessary trials. In chapter 7, we suggested that it may be important to model the variation of the spectra in time. We saw that to do this by remembering previous spectral frames in the states of the underlying Markov models meant using models that have too many free parameters. This led to the idea of modeling time-derivative information by adjoining adjacent spectral vectors. Unfortunately, it is not clear how to model such information as straightforwardly in an LPC waveform model of the type we have been discussing. Another problem is that the error terms are likely to be correlated and this correlation is not modeled. Since the models here require the computation of 200 times as many output probabilities per second as the standard IBM models, this correlation problem is likely to be far more serious than in with the standard models. If the acoustic signal is disrupted for a brief amount of time by some noise event, for example, it may effect only a couple of the standard model's centisecond-long frames, but will effect a few hundred waveform samples. Finally, there is the inadequacy of assuming the speech can be modeled as the output of a series of lossless acoustic tubes. There are a number of problems with this assumption. One commonly mentioned example is the inability of such a model to account for zeros in the transfer function that can be introduced by the nasal cavity.

In summary, although modeling speech waveforms directly on a sample-by-sample basis has some extremely attractive features, the simple LPC-based hidden Markov models described above have a number of serious problems. The further assessment and evaluation of possible solutions to these problems will involve conducting a large number of waveform-model experiments. It is unfortunate that the computational resources for these experiments do not now exist. But as we look forward to a future with faster and faster computers, we can also look forward to the possibility of speech recognition systems in which all components are uniformly integrated into one elegant statistical engine.

## 10. Implementation

### 10.1 Thresholding the Forward-Backward Computation

In this section, we will examine a method of reducing the amount of computation in MLE and MMIE for hidden Markov models by thresholding the forward-backward computation. This computation consists of two passes over the data. In the forward pass, for each time,  $t$ , and for each state,  $i$ , we compute  $\alpha_i(t)$ , the probability of generating the sequence  $y_1^t$ , and ending up in state  $i$  at time  $t$ . In the backward pass, for each time,  $t$ , and for each state,  $i$ , we compute  $\beta_i(t)$ , the probability of generating the sequence  $y_{t+1}^T$ , having started in state  $i$ . We saw that the  $\alpha$ 's at time  $t$  can be easily computed from the  $\alpha$ 's at time  $t-1$  by taking advantage of the following recursion:

$$\alpha_i(t) = \sum_j \alpha_j(t-1) a_{ji} b_{ji}(y_t), \quad (10.1)$$

where  $a_{ji}$  is the prior probability of making the transition from state  $j$  to state  $i$ , and  $b_{ji}(y_t)$  is the probability of generating the observation  $y_t$  from the output distribution associated with the transition  $j \rightarrow i$ . Similarly the  $\beta$ 's at time  $t$  can be easily computed from the  $\beta$ 's at time  $t+1$  by taking advantage of the following recursion:

$$\beta_i(t) = \sum_j a_{ij} b_{ij}(y_{t+1}) \beta_j(t+1). \quad (10.2)$$

In both MLE and MMIE the  $\alpha$ 's and the  $\beta$ 's are computed in order to compute, for every pair of states  $i$  and  $j$ , and for  $t = 1$  to  $T$ ,  $\gamma_{ij}(t)$ , the probability of having made the transition  $i \rightarrow j$  at time  $t$  given that the sample  $y_1^T$  was generated by the model. As is described in Chapter 3, these  $\gamma$ 's serve as coefficients of terms which are summed together in various ways. The resultant accumulations are then used to compute reestimates of the model parameters.

If certain  $\gamma$ 's become very small relative to other  $\gamma$ 's, terms associated with these small  $\gamma$ 's will in general not affect the final reestimates, or at least they will not affect the objective function as computed with the reestimated model. As an example of how the reestimates may be affected but not the objective function consider the following situation. Suppose that it is very unlikely to ever visit a particular state,  $i$ . The  $\gamma$ 's associated with all transitions from  $i$  will then be relatively small. Reestimated probabilities of the transitions

from this state certainly will depend on these small  $\gamma$ 's, but since the state is rarely visited these transition probabilities will have a very minor effect on the performance of the model on the training sample.

Since an  $\alpha$  appears as a factor in each  $\gamma$ , if during the course of the forward pass we notice that certain  $\alpha$ 's are becoming very small relative other  $\alpha$ 's, we can simply assume that these small  $\alpha$ 's are zero without significantly affecting the performance of the model which is derived by our training procedure, be it MLE or MMIE. An efficient way of implementing this thresholding is to set to zero all  $\alpha$ 's at each time,  $t$ , which are significantly less than the maximum  $\alpha$  at  $t$ . Define  $\kappa(t)$  to be the largest alpha at time  $t$ ,

$$\kappa(t) \triangleq \max_i \alpha_i(t). \quad (10.3)$$

Then, given a threshold  $\tau$ , for each  $i$  such that  $\alpha_i(t) < \tau \kappa(t)$ , set  $\alpha_i(t)$  equal to zero before moving on to compute the  $\alpha$ 's at time  $t+1$ . At time  $t+1$  only those  $\alpha$ 's from time  $t$  which are greater than zero will enter into the computation. Those which have been set to zero can be ignored. If the state-space of the underlying Markov chain is large, this thresholding will be very important in reducing the computation to a manageable size. This technique, which is implemented in the IBM system, is essentially a variant of the beam search as performed by the Harpy system, [Lowerre 76].

The backward pass can be thresholded in the same manner. In addition, we can further threshold this pass by noticing that if the probability of generating  $y_1^t$  and arriving in state  $i$  at time  $t$  is very small, then it is irrelevant for our purposes what the probability of generating  $y_{t+1}^T$  having started in state  $i$  at time  $t$  is. Therefore, we need only compute  $\beta_i(t)$  on the backward pass if  $\alpha_i(t)$  was not zero, or not set to zero, on the forward pass.

The appropriate value of the threshold  $\tau$  must be determined empirically. If  $\tau$  is too large, more computation than necessary will be performed. If  $\tau$  is too small, paths which contribute significantly to the probability of generating  $y_1^T$  will be prematurely thresholded out of consideration. If this happens, the probability of generating  $y_1^T$ , as computed on the forward pass, will be inaccurate, the conditional expectations used in the reestimates in the forward-backward algorithm will be inaccurate, and the derivatives of the probability of generating  $y_1^T$  used in MMIE will be inaccurate. One easily detectable example of such over-thresholding occurs when a model is used in which not all states are final states. With such a model, if  $\alpha_i(T)$  is zero for every final state,  $i$ , then clearly the forward-pass computation has been over-thresholded. Similarly, if on the backward pass  $\beta_i(0)$  is zero for every initial state,  $i$ , then the backward-pass computation has been over-thresholded. If

neither the forward pass nor the backward pass was over-thresholded, then the probability of generating  $y_1^T$  as computed on the forward pass should be equal to the probability of generating  $y_1^T$  as computed on the backward pass.

Let us examine this problem of over-thresholding in more detail. Suppose the probability of generating an observation  $y_t$  from any output distribution associated with a transition to state  $i$  is small when compared to the probability of generating  $y_t$  from distributions associated with other transitions. Then, all else being equal, paths which arrive at state  $i$  at time  $t$  will tend to be less probable, given  $y_1^T$ , than other paths. But suppose, as is usually the case, that all else is not equal. In particular the probability of generating  $y_{t+1}^T$  having started in state  $i$  may be much greater than the probability of generating  $y_{t+1}^T$  having started in any other state. Then in the thresholding process we are likely to set  $\alpha_i(t)$  to zero because we have not yet examined  $y_{t+1}^T$ . As a result, we may end up setting most of the probability of generating  $y_1^T$  to zero.

This danger is particularly acute when modeling observations from a continuous space. As we observed in Chapter 5, the tails of a Gaussian drop off very rapidly, and as a result, output densities at many observations in a sequence can be extremely small. Furthermore, when using continuous-parameter models, it is often the case that in the sequence of output probabilities along the most probable path given  $y_1^T$ , there are a few extremely small values. This is because there are usually even more such small values along less probable paths. Empirically, this phenomenon is often so severe when modeling observations from a continuous space that in order to avoid over-thresholding,  $\tau$  must be greater than the dynamic range of the floating-point representation of numbers used by the system. For example, in the IBM 370 architecture positive floating-point numbers range from  $10^{-78}$  to  $10^{75}$ . In many continuous-parameter hidden Markov model experiments this dynamic range of 154 orders of magnitude has been found to be too small. It is not simply that the probability of generating a sequence becomes smaller and smaller as the length of the sequence grows; this problem can be easily circumvented by normalizing the  $\alpha$ 's and  $\beta$ 's after all the states at a particular time have been processed. Rather, it is that certain  $\alpha$ 's, or  $\beta$ 's, which we do not want to be zero at a particular time, become very small relative to other  $\alpha$ 's, or  $\beta$ 's, at that time. Even if there is no explicit thresholding, unless an alternative representation of probabilities is used, the dynamic range of the system can impose an implicit threshold which is often too severe.

## 10.2 Representing Probabilities by Their Logarithms

Consider the criteria we would like of an alternative representation of the probabilities, or probability densities in the forward-backward computation. First, since we are representing probabilities, we only need to represent positive numbers. Second, as we have just seen, the dynamic range of whatever representation we choose must be large, on the order of 1000 orders of magnitude. Third, we do not require a lot of precision in the representation of our probabilities, because the algorithms used in both MLE and MMIE are extremely well conditioned. If, during the course of the computation, probabilities are off by a few percent, it will normally only alter the final parameter estimates by very small amounts. Finally, because the estimation of parameters for models of speech with either MLE or MMIE can be computationally very expensive, it is important that we be able to perform the adds, multiplies, and compares in the forward-backward computation rapidly.

A simplified version of the sign/logarithm number system presented by Swartzlander and Alexopoulos meets all of these criteria [Swartzlander 75]. In their system, numbers are represented by a sign bit and an integer logarithm. Since we do not have to represent negative numbers, we can dispense with the sign bit. In this system, a positive number,  $x$ , will be represented by  $\omega(x)$ , an integer approximation to the logarithm of  $x$  to a base  $\xi$ :

$$\omega(x) \triangleq \left\lfloor \log_{\xi} x + \frac{1}{2} \right\rfloor. \quad (10.4)$$

Using this definition,

$$\left| \omega(x) - \log_{\xi} x \right| \leq \frac{1}{2}. \quad (10.5)$$

If, for example,  $\xi = 1.01$ , then  $\xi^{\omega(x)}$  will be within one half of one percent of  $x$ . If  $\omega(x)$  is an  $n$ -bit integer, this representation will have a dynamic range of  $2^n \log_{10} \xi$  orders of magnitude. If  $n = 32$  and  $b = 1.01$ , there will be a dynamic range of over one million orders of magnitude.

Let  $x$  and  $y$  be two positive numbers. Let the integers  $k_x$  and  $k_y$ , and the numbers  $\epsilon_x$  and  $\epsilon_y$  be such that

$$\left| k_x - \log_{\xi} x \right| \leq \epsilon_x, \quad \text{and} \quad \left| k_y - \log_{\xi} y \right| \leq \epsilon_y. \quad (10.6)$$

Since

$$\xi^{k_x} \xi^{k_y} = \xi^{k_x + k_y}, \quad (10.7)$$

we define multiplication in this representation as

$$k_x \otimes k_y \triangleq k_x + k_y. \quad (10.8)$$



Multiplication can thus be implemented with one integer add instruction. Since errors accumulate in the logarithm of the product

$$|(k_x \otimes k_y) - \log_{\xi} xy| \leq \epsilon_x + \epsilon_y. \quad (10.9)$$

Addition is slightly more complicated. Define the function  $\psi$  as

$$\psi(n) \triangleq \omega(1 + \xi^n). \quad (10.10)$$

Since

$$\xi^{k_x} + \xi^{k_y} = \xi^{k_x} (1 + \xi^{k_y - k_x}), \quad (10.11)$$

we define addition in this representation as

$$k_x \oplus k_y \triangleq k_x \otimes \psi(k_y - k_x). \quad (10.12)$$

The function  $\psi$  can be implemented by table lookup. If  $\xi > 1$ , then as  $n \rightarrow -\infty$ ,  $(1 + \xi^n) \rightarrow 1$ , and  $\psi(n) \rightarrow 0$ . In fact, if  $(\log_{\xi}(1 + \xi^n) + \frac{1}{2}) < 1$ , then  $\psi(n) = 0$ . If we redefine  $\oplus$  as

$$k_x \oplus k_y \triangleq \begin{cases} k_x \otimes \psi(k_y - k_x), & \text{if } k_y \leq k_x, \\ k_y \otimes \psi(k_x - k_y), & \text{otherwise,} \end{cases} \quad (10.13)$$

then the argument to  $\psi$  will always be nonpositive, and we can reduce the size of the table needed to implement  $\psi$ . Table 10.1 lists table sizes for different values of  $\xi$ . Addition can thus be implemented with one integer add instruction, one subtract, two compares, and one table lookup. Since the table lookup adds an additional round-off error to the logarithm of the sum,

$$|(k_x \oplus k_y) - \log_{\xi}(x + y)| \leq \frac{1}{2} + \max(\epsilon_x, \epsilon_y). \quad (10.14)$$

$\xi$	Table Size
1.01	533
1.001	7,606
1.0001	99,041
1.00001	1,220,614

Table 10.1:  $\psi$  Lookup Table Sizes

In the experiments described in the next chapter, this representation system was used with  $\xi = 1.0001$ , and with the logarithms stored as 32-bit integers.

### 10.3 Maximum Mutual Information Estimation

In this section, we will examine the implementation of MMIE in speech recognition. Given a language model,  $\ell$ , an acoustic parameter sequence,  $\mathbf{y} = \mathbf{y}_1^T$ , and corresponding word sequence,  $\mathbf{w} = \mathbf{w}_1^n$ , the goal of MMIE is to choose a vector of parameters,  $\Theta$ , which enables a family of Markov models to maximize an estimate of the mutual information between  $\mathbf{w}$  and  $\mathbf{y}$ ,

$$f_{\text{MMIE}}(\Theta) = \log \Pr_{\Theta}(\mathbf{Y} = \mathbf{y} \mid \mathbf{W} = \mathbf{w}) - \log \Pr_{\ell, \Theta}(\mathbf{Y} = \mathbf{y}). \quad (10.15)$$

The probabilities in (10.15) can be computed by assigning a hidden Markov model to each word sequence  $\mathbf{w}$ . For example, if  $m$  is the model corresponding to the word sequence  $\mathbf{w}$ , then

$$\Pr_{\Theta}(\mathbf{Y} = \mathbf{y} \mid \mathbf{W} = \mathbf{w}) = \Pr_{\Theta}(\mathbf{Y} = \mathbf{y} \mid M = m). \quad (10.16)$$

$\Pr_{\ell, \Theta}(\mathbf{Y} = \mathbf{y})$  is computed by summing over all possible word sequences

$$\Pr_{\ell, \Theta}(\mathbf{Y} = \mathbf{y}) = \sum_{\tilde{\mathbf{w}}} \Pr_{\Theta}(\mathbf{Y} = \mathbf{y} \mid \mathbf{W} = \tilde{\mathbf{w}}) \Pr_{\ell}(\mathbf{W} = \tilde{\mathbf{w}}). \quad (10.17)$$

We can drastically reduce the computation involved in computing the derivative of  $f_{\text{MMIE}}(\Theta)$  by making two assumptions. First, assume that the training data  $\mathbf{y}$  can be segmented into  $n$  subsequences,  $\mathbf{y}^{(1)}, \dots, \mathbf{y}^{(n)}$ , one  $\mathbf{y}^{(i)}$  for each  $w_i$  in  $\mathbf{w}_1^n$ , and that for any word sequence  $\tilde{\mathbf{w}}_1^m$ ,

$$\Pr_{\Theta}(\mathbf{Y} = \mathbf{y} \mid \mathbf{W} = \tilde{\mathbf{w}}_1^m) = \begin{cases} \prod_{i=1}^m \Pr_{\Theta}(\mathbf{Y}^{(i)} = \mathbf{y}^{(i)} \mid W_i = w_i), & \text{if } m = n; \\ 0, & \text{otherwise.} \end{cases} \quad (10.18)$$

Second, assume that for any word sequence  $\tilde{\mathbf{w}}_1^m$ ,

$$\Pr_{\ell}(\mathbf{W} = \tilde{\mathbf{w}}_1^m) = \prod_{i=1}^m \Pr_{\ell}(W_i = \tilde{w}_i). \quad (10.19)$$

Using these two assumptions, we rewrite  $\Pr_{\ell, \Theta}(\mathbf{Y} = \mathbf{y})$  as

$$\Pr_{\ell, \Theta}(\mathbf{Y} = \mathbf{y}) = \sum_{\tilde{\mathbf{w}}_1^n} \prod_{i=1}^n \Pr_{\Theta}(\mathbf{Y}^{(i)} = \mathbf{y}^{(i)} \mid W_i = \tilde{w}_i) \Pr_{\ell}(W_i = \tilde{w}_i). \quad (10.20)$$

Furthermore, since the sum is over all possible word sequences, and since every word can appear in every position,

$$\begin{aligned} \Pr_{\ell, \Theta}(\mathbf{Y} = \mathbf{y}) &= \prod_{i=1}^n \sum_{\tilde{w}} \Pr_{\Theta}(\mathbf{Y}^{(i)} = \mathbf{y}^{(i)} \mid W_i = \tilde{w}) \Pr_{\ell}(W_i = \tilde{w}) \\ &= \prod_{i=1}^n \Pr_{\ell, \Theta}(\mathbf{Y}^{(i)} = \mathbf{y}^{(i)}). \end{aligned} \quad (10.21)$$

Each segment of speech can now be handled separately since

$$f_{\text{MMIE}}(\theta) = \sum_{i=1}^n \left[ \log \Pr_{\theta} (Y^{(i)} = y^{(i)} \mid W_i = w_i) - \log \Pr_{L, \theta} (Y^{(i)} = y^{(i)}) \right]. \quad (10.22)$$

The first assumption, equation (10.18), is certainly not a good one in continuous speech. It ignores errors which occur across word boundaries. For example, commonly in continuous speech recognition a word like *opportunities* is misrecognized as the pair of words *opportunity is*. In equation (10.18), it is assumed that an acoustic sequence either sounds like *opportunities* or *opportunity is*, but not like both. Another problem with (10.18) is that it prevents the modeling of coarticulation across word boundaries. Suppose  $w$  consists of the word sequence *gas shortage*. It is very likely that a speaker will pronounce  $w$  as *gash shortage*. In this situation equation (10.18) will not be accurate, because the probability that whatever is deemed to be the first acoustic subsequence,  $y^{(1)}$ , is an acoustic realization of the word *gas* very much depends on the identity of the following word. If the following word was *station*, for example, it would be very unlikely that *gas* was pronounced as if the speaker was saying *gash*. In the isolated-word experiments described in this thesis, these types of problems were assumed to be minor, and (10.18) was assumed to be true in order to simplify the implementation of MMIE. Acoustic subsequences were determined from a Viterbi alignment of the training data with the training script.

The second assumption, (10.19), is also certainly not true, at least of any natural language like English. However, it is not clear that even if we could tackle the computational problems associated with a complicated language model, we would want to. In general, the size of the training script used when estimating acoustic-model parameters is far less than that needed to reliably estimate language-model parameters. As a result, we will not be able to hope to correct for deficiencies in a language model in the estimation of acoustic-model parameters. Normally, acoustic-model parameters will be estimated from the acoustic realization of a script of about 1000 words. In a large-vocabulary system, this will not even be enough words to ensure that each word in the recognizer's vocabulary occurs in the training script. Ideally, we would like to estimate acoustic parameters so that the resultant models discriminate well between words that are likely to occur in the same linguistic contexts. If only a small fraction of the words in the recognizer's vocabulary are found in the training script, we must hope to do this by estimating acoustic parameters so that phones, or some other sort of sub-word units that are shared among words, are discriminated from other phones. In such a situation, it is not at all clear which language model probabilities will ensure that acoustic parameters are chosen so that phones are

discriminated from other phones to the appropriate extents. By using a unigram language model, as in equation (10.19), and by assuming that each phone occurs in many words in the training script, we can hope that the importance assigned to discriminating between a pair of phones will be roughly approximated by the product of the unigram probabilities of the phones. Actually, we can hope for even more, since when parameters are estimated to discriminate words from other words, it will be important to discriminate between different phones that occur in similar acoustic contexts.

A better alternative to using a static unigram model in equation (10.21) is to vary the unigram probabilities according to the words in the training script. At each position,  $i$ , we can average over all linguistic contexts in which the word  $w_i$  can appear to determine varying unigram language model probabilities which can then be used in the expression of  $\text{Pr}_{\ell, \Theta}(Y = y)$ . For (10.21) we would use

$$\text{Pr}_{\ell, \Theta}(Y = y) = \prod_{i=1}^n \sum_{\tilde{w}} \text{Pr}_{\Theta}(Y^{(i)} = y^{(i)} | W_i = \tilde{w}) \hat{\text{Pr}}_i(W_i = \tilde{w}), \quad (10.23)$$

where  $\hat{\text{Pr}}_i(W_i = \tilde{w}_i)$ , is the probability that  $\tilde{w}_i$  occurs in a context in which  $w_i$  can occur. The  $\hat{\text{Pr}}_i(W_i = \tilde{w}_i)$ 's can be precomputed from  $\ell$ . By doing this, we will, in essence, be pretending that we had a large enough training sample to have seen every word, in every linguistic context, the appropriate number of times.

Let us consider one final simplifying assumption. In practice, for each acoustic segment,  $y^{(i)}$ , there is a relatively small set of words  $\Lambda_i$ , which contribute significantly to  $\sum_{\tilde{w}} \text{Pr}_{\ell}(W_i = \tilde{w}) \text{Pr}_{\Theta}(Y^{(i)} = y^{(i)} | W_i = \tilde{w})$ , and it reduces the computation in MMIE to assume that

$$\begin{aligned} & \sum_{\tilde{w}} \text{Pr}_{\ell}(W_i = \tilde{w}) \text{Pr}_{\Theta}(Y^{(i)} = y^{(i)} | W_i = \tilde{w}) \\ &= \sum_{\tilde{w} \in \Lambda_i} \text{Pr}_{\ell}(W_i = \tilde{w}) \text{Pr}_{\Theta}(Y^{(i)} = y^{(i)} | W_i = \tilde{w}). \end{aligned} \quad (10.24)$$

Each set,  $\Lambda_i$ , can either be precomputed from a list of words confusable with  $w_i$ , or can be computed by examining  $y^{(i)}$ . When estimating the detailed-match parameters for a large-vocabulary system, it is convenient to determine the  $\Lambda_i$ 's from lists of words produced by the fast match.

## 11. E-Set Experiments

### 11.1 Previous Results

This chapter describes a series of experiments testing the ideas which have been discussed in previous chapters. It was important to find a small task for these experiments, since many of these techniques involve a large amount of computation. It was also important, however, that the task be difficult enough for there to be room for improvement. The task chosen was the isolated-word, multi-speaker recognition of the *e*-set, the letters *b*, *c*, *d*, *e*, *g*, *p*, *t*, *v* and *z*. The task is small since it is an isolated-word task with a vocabulary of nine one-syllable words. Yet it is also difficult: the standard IBM system has a performance of only 79 percent on this task.

A number of other groups have also experimented on the *e*-set. While the tasks and the data used by the different groups differed in important ways, making it difficult to assess the relative merits of the techniques used by these groups, it is nonetheless of interest to briefly review their results.

Rabiner and Wilpon [Rabiner 79] examined the performance of a dynamic time warping system on a 39-word task which included the *e*-set. Their purpose was to investigate clustering techniques as a means of determining the templates for their system. They recorded 100 talkers, 50 male and 50 female, speaking their 39-word vocabulary. Words were spoken in isolation in a soundproof booth over dialed-up phone lines. They used 3 sets of test data. The first two sets consisted of samples from speakers who were not in the original group of 100. The third set consisted of new samples from speakers who were in the original 100. In discussing their results, we will merge the results from these three test sets. In addition to the *e*-set their vocabulary contained other letters, the digits, and the words *stop*, *error* and *repeat*. In the three test sets taken together, there were 252 samples from the *e*-set. Of these 252 samples, only 6 were misrecognized as any of the 30 words not in the *e*-set. If we exclude these 6 words, they recognized 56 percent of the words in the *e*-set correctly. While this might seem like a relatively poor result, it is important to keep in mind that the 3200 Hz cutoff in telephone speech makes it particularly difficult to distinguish English consonants from one another.

In 1981, Waibel and Yegnanarayana [Waibel 81] experimented with a variety of dynamic time warping algorithms on the *e*-set. Their task was a speaker-dependent, isolated-word

task, and their data was recorded with a close-talking lip mike. The average performance across speakers of their best warping algorithm was approximately 60 percent. Using the same algorithm to recognize the ten digits, they obtained a performance of 98 percent. The difference in these two numbers demonstrates that the words in the *e*-set are extremely confusable with one another.

An improvement to classical dynamic time warping results on the *e*-set was obtained in 1982 by Bradshaw, Cole and Li [Bradshaw 82]. They investigated a variant of a technique suggested by Rabiner and Wilpon [Rabiner 81]. Utterances were first classified into confusable classes, and then recognized with an algorithm which assigns weights to frames so as to discriminate between words within a given class. They evaluated their algorithm on a task with a vocabulary of the *e*-set and the word *three*. Utterances were spoken in isolation using a close-talking lip mike. 8 speakers, 4 male and 4 female, uttered 5 instances of each word for training data, and an additional 5 instances for test data. Using their two-pass approach they improved system performance from 63 percent to 84 percent. After having obtained this improvement, they augmented their system with a module which produced a set of linguistically motivated features for each utterance. By treating these features as an additional frame in their recognizer, they were able to increase the performance of their system to 90 percent.

In subsequent years, the speech recognition group at Carnegie-Mellon has continued to explore this feature-based approach to speech recognition. They removed the template matching module from their system entirely, and constructed a recognizer built around a decision tree. At each node in the tree, the recognizer asks a question about the features which have been extracted from the utterance to be recognized. It then uses the answer to this question to classify the utterance into one of a few disjoint sets of words. Associated with each leaf of the tree is a set consisting of a single word. They have tested this recognizer on a speaker-independent isolated-word *e*-set task using lip-mike data, and obtained recognition rates of 83, 88 and 90 percent [Lasry 84], [Cole 86], [Stern 87].

## 11.2 Data

This section describes the data that was used in the experiments that are presented in this chapter.

One hundred male speakers were recorded, each speaking the 26 letters in the alphabet twice, once for training data, and once for test data. The 26 letters were spoken in 3 random sentences, so that each speaker spoke 6 sentences. Speakers were instructed to leave brief pauses between the words in their sentences. Due primarily to speaking errors, a fraction of the sentences had to be discarded. After the words in the *e*-set were extracted from the remaining sentences, there were 836 words of training data and 886 words of test data.

The recordings were made in offices and labs at the IBM Thomas J. Watson Research Center in Yorktown Heights, New York. The speech was recorded with a pressure-zone microphone which was mounted next to the screen of a PC-AT. It was digitized in real time at 20,000 Hz. No effort was made to suppress the noise from the fans and disks on the PC or from other equipment in a speaker's work environment. As a result the data is relatively noisy. An estimate of the signal-to-noise ratio for this data was made by first using a Viterbi alignment to assign each 10-centisecond frame of speech to a phonetic category, including a noise category, and then by computing the ratio of the average signal power in each of the categories to that in the noise category. The fifteen phonetic categories used are listed in Table 11.1. Table 11.2 lists the spellings of the words in the *e*-set in terms of the categories. In Table 11.3 the signal-to-noise ratio in decibels for each of the categories is presented. The relatively low ratio for the D phone is probably due to *d*-speech being misaligned with the E phone. If we average over all frames which were not mapped into an onset or trail-off category, we obtain an average signal-to-noise ratio of 16.4 decibels. We see, then, that this data is very noisy when compared with normal lip-mike data, which has a signal-to-noise ratio of about 50 decibels.

<i>Name</i>	<i>Description</i>
Noise	noise
OnB	onset of voiced consonant such as <i>b</i>
OnE	onset of vowel in <i>e</i>
OnS	onset of sibilant such as <i>s</i>
OnV	onset of consonant in <i>v</i>
B	consonant in <i>b</i>
D	consonant in <i>d</i>
J	consonant in <i>g</i>
P	consonant in <i>p</i>
S	consonant in <i>s</i>
T	consonant in <i>t</i>
V	consonant in <i>v</i>
Z	consonant in <i>z</i>
E	vowel in <i>e</i>
TrE	trail-off of the vowel in <i>e</i>

Table 11.1: Phonetic Categories for E-Set Experiments

<i>Word</i>	<i>Phonetic Representation</i>
<i>b</i>	Noise OnB B E TrE Noise
<i>c</i>	Noise OnS S E TrE Noise
<i>d</i>	Noise OnB D E TrE Noise
<i>e</i>	Noise OnE E TrE Noise
<i>g</i>	Noise OnB D J E TrE Noise
<i>p</i>	Noise P E TrE Noise
<i>t</i>	Noise T E TrE Noise
<i>v</i>	Noise OnV V E TrE Noise
<i>z</i>	Noise OnS Z E TrE Noise

Table 11.2: Phonetic Spellings for E-Set Experiment



*Phonetic Category    Signal-to-Noise Ratio*

E	17.5
TrE	7.0
OnB	4.3
B	17.5
D	5.9
OnE	7.6
OnV	3.7
V	14.7
OnS	4.5
S	18.8
J	15.9
P	13.1
T	15.4
Z	15.9

**Table 11.3:** Signal-to-Noise Ratios in Decibels**11.3 Models and Methods**

In this section we describe the word models and certain details of the methods used in the series of experiments performed on this *e-set* data.

The speech was processed with a slight variant of the standard IBM signal-processing algorithm that is described in Chapter 4. The only modification was to perform the long-term adaptation separately for each speaker. This long-term adaptation was performed in two passes. In the first pass, long-term signal averages were computed from all 26 letters uttered by the speaker, and then used in a second pass to extract the ear-model parameters. The training and test data were, of course, processed separately. The acoustic labels used in the discrete-parameter experiments were obtained by the standard IBM labeling program that is also described in Chapter 4. In the continuous-parameter experiments, parameter vectors that had not already been projected onto linear discriminants were rotated with principal components in an attempt to make the off-diagonal terms in the covariance matrices as small as possible. This was important when diagonal Gaussian distributions were used. It also affected the convergence rate of the gradient search methods used in MMIE.

Except where noted, word models were constructed by concatenating phonetic models

according to the phonetic spellings in Table 11.2. The phonetic models were of three varieties. All onset phones were modeled with finite-state machines like the one shown in Figure 11.1. The trail-off phone, TrE, was modeled with a machine similar to that used for onset phones, except that each of the three output distributions was allowed to occur up to four as opposed to two times. All other phones were modeled with machines like the one shown in Figure 11.2. Since each of the 15 phone models contained three different output distributions, there were a total of 45 distinct output distributions.

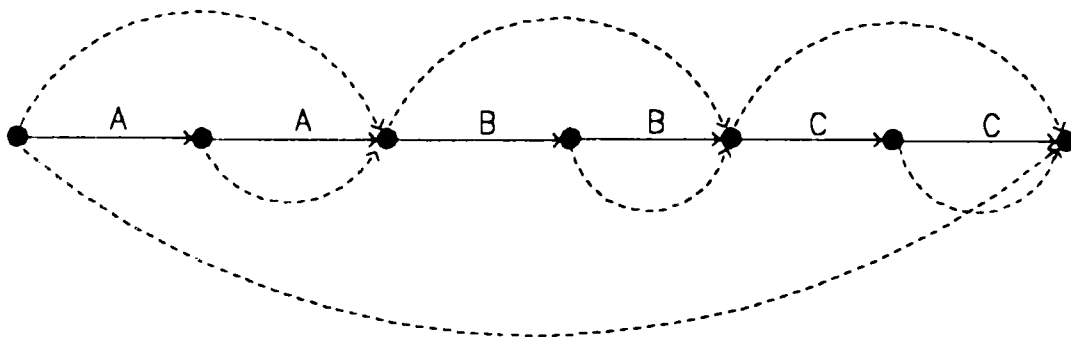


Figure 11.1: Onset Phone Machine

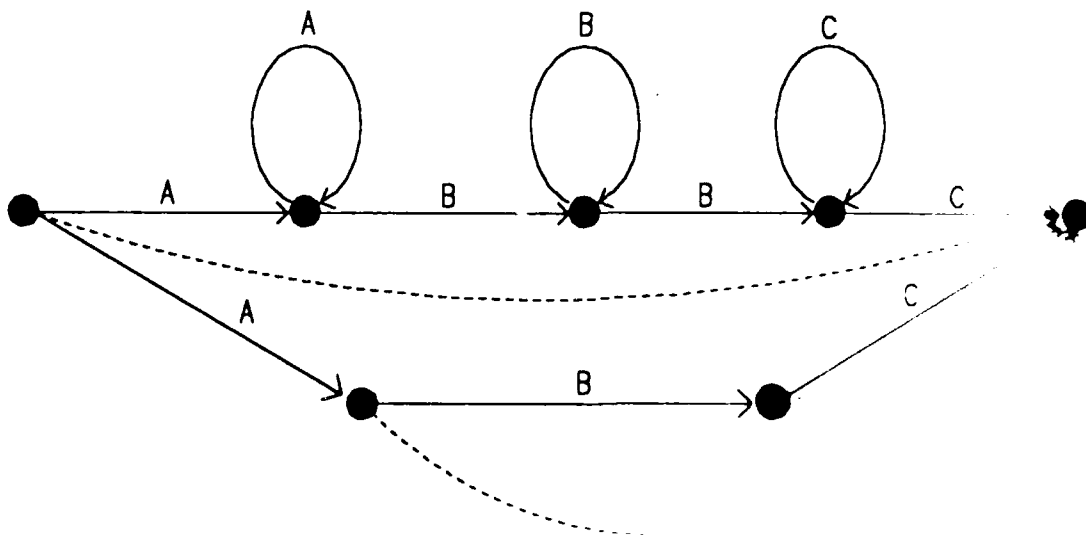


Figure 11.2: Normal Phone Machine

AD-A188 529

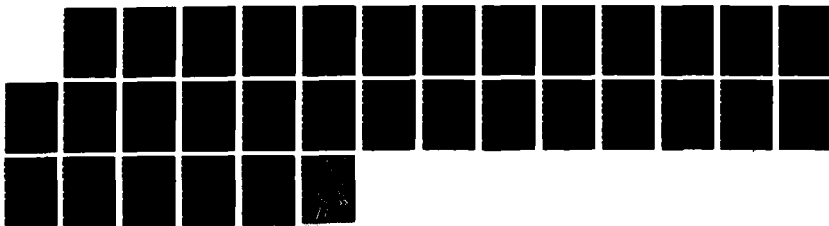
THE ACOUSTIC-MODELING PROBLEM IN AUTOMATIC SPEECH  
RECOGNITION(U) CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT  
OF COMPUTER SCIENCE P F BROWN DEC 87 CMU-CS-87-125  
AFMAL-TR-87-1161 F33615-84-K-1520

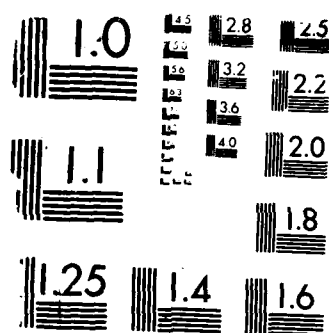
2/2

UNCLASSIFIED

F/G 25/4

NL





MICROCOPY RESOLUTION TEST CHART

Hidden Markov model parameters were estimated with both MLE and with MMIE. MLE was performed by running the forward-backward algorithm for 6 iterations. At the end of these 6 iterations, it was always observed that the probability of generating the test data was increasing by less than one percent per iteration. This is by no means a guarantee that the algorithm is close to convergence. Empirically, however, it is usually the case that the size of the improvement in the likelihood decreases with successive iterations.

MMIE was performed by gradient search starting at a vector of parameters derived by MLE. The step size of the hill-climbing algorithm was adjusted automatically using an ad-hoc method which is based on an assumption that the magnitude of the projection of the gradient onto the last direction stepped is a quadratic function. Suppose the hill-climbing routine had just taken a step from the parameter vector  $\tilde{\theta}$  to the parameter vector  $\theta$ . Denote the gradient of the objective function,  $f$ , at  $\tilde{\theta}$  by  $\nabla f(\tilde{\theta})$ , and similarly the gradient at  $\theta$  by  $\nabla f(\theta)$ . If  $f(\theta) < f(\tilde{\theta})$ , then the step size was reduced by a factor of three, and the next step was taken from  $\tilde{\theta}$  in the direction of  $\nabla f(\tilde{\theta})$ . If, on the other hand, the objective function improved on the last step, the next step was taken from  $\theta$  in the direction of  $\nabla f(\tilde{\theta})$ . The step size for such a step was determined by first computing the magnitude of  $\nabla f(\tilde{\theta})$ , and the magnitude of the projection of  $\nabla f(\theta)$  onto  $\nabla f(\tilde{\theta})$ . If the projection of  $\nabla f(\theta)$  onto  $\nabla f(\tilde{\theta})$  was greater than the magnitude of  $\nabla f(\tilde{\theta})$  itself, then the previous step size was doubled. Otherwise, it was assumed that the magnitude of the projection of the gradient onto  $\nabla f(\tilde{\theta})$  was a quadratic function of the distance stepped, and the step size was adjusted to be equal to the distance to the minimum of this quadratic function, except that it was never altered by more than a factor of two on any iteration. The hill-climbing was terminated when for each of ten consecutive iterations there was less than a two-percent improvement in the objective function. This normally occurred after about 50 to 100 iterations.

Once the model parameters had been trained, test data was decoded using a unigram language model with a uniform distribution over the the nine words. Since each word occurred either 98 or 99 times in the test data, this language model was nearly perfectly accurate.

#### 11.4 Resources

In this section we briefly describe the resources that were used to conduct the *e*-set experiments.

These experiments were conducted on an IBM 3090. The 3090 runs IBM 370 assembly language with a cycle time of 18 nanoseconds. It is Whetstone-benchmarked at 14 MIPS. It runs the Dongarra LINPACK routine at 7 megaflops. The 3090 that was used for these experiments was also equipped with an IBM 3090 Vector Facility. This vector processor runs the Dongarra LINPACK at 12 megaflops.

The IBM CMS operating system was used. All programs ran within an 8-megabyte address space.

The quadratic forms in full-covariance distributions were computed with programs written in FORTRAN for the vector processor. All other routines were written in PL.8, and run in scalar mode. PL.8 is a PL/I derivative for which there is an extremely efficient compiler. There are approximately 10,000 lines of PL.8 and 100 lines of FORTRAN.

The number of CPU hours it takes to run an experiment is very much determined by the type of output distributions, the type of parameter estimation, the dimension of the parameter vectors in continuous-parameter experiments, and by the number of transitions in the word models.

Discrete output probabilities can be computed by table lookup, and are, therefore, essentially free. It takes approximately  $n$  adds and  $2n$  multiplies to compute an  $n$ -dimensional diagonal Gaussian density. It takes approximately  $n^2$  adds and  $.5n^2$  multiplies to compute an  $n$ -dimensional full-covariance Gaussian density. A similar amount of time is required to compute the density of a Richter mixture of full-covariance Gaussians.  $n$ -dimensional conditional Gaussian densities can be computed with approximately  $2.5n^2$  multiply-adds. Each MLE accumulation, for a given frame and for a given distribution, involves roughly the same amount of computation as the evaluation of a probability density for the same distribution. Each MMIE accumulation involves roughly three times as much computation as a density evaluation.

It takes roughly twice as much computer time to perform the forward-backward computation which is used in MLE and MMIE as it does to perform the forward-pass which is used in decoding. In MLE, the forward-backward computation is only performed for the correct word sequence. When decoding, unless an initial fast-match is used, the forward pass must be run for every possible word sequence. Since words were processed in isolation,

and since the *e*-set task involves a nine-word vocabulary, the forward pass was performed nine times during decoding for each word of speech. MMIE estimation was performed by first decoding each word of speech, and then by using the forward-backward computation to evaluate derivatives for all words with a forward-pass score that was within one percent of the forward-pass score for the correct word. If this list only included the correct word, then it was not necessary to perform any forward-backward computations. On average, one MMIE iteration required roughly as much computer time as a decoding iteration and an MLE iteration together. In both MLE and MMIE, the computer time involved in the actual updating of the parameters was negligible in comparison with the time it took to accumulate the statistics from which the updates were made.

Decoding, MLE and MMIE are all approximately linear in the number of transitions in the word models, and are linear in the number of frames of speech data. The actual topology of the word models is also very important, but it is difficult to give a rule of thumb which will help to estimate computer time as a function of word-model topology. The word models described in the previous section were used in all of the experiments discussed in this chapter. There were 110,000 frames of training data and 115,000 frames of test data.

Accurate records of the amount of computer time to run each experiment were not kept. The following table is based on the author's recollections of approximately how long it took to run a few typical experiments. The continuous-parameter experiments in the table are for hidden Markov models which used 20-dimensional full-covariance Gaussian distributions.

<i>Task</i>	<i>3090 CPU Hours</i>
Discrete Decoding	less than 1
Discrete MLE	less than 1
Discrete MMIE	15
Continuous Decoding	1
Continuous MLE	2
Continuous MMIE	70

**Table 11.4:** Approximate IBM-3090 CPU Hours for Experiments

### 11.5 Results

In this section we examine the results of the experiments run on the *e*-set data described in Section 11.2. A complete list of the results from this series of experiments can be found in Table 11.7. The abbreviations appearing in this table are defined in Table 11.6. In the discussion which follows, I will refer to experiments by the experiment numbers in Table 11.7.

First, a benchmark experiment was performed using discrete output distributions and observations which were selected from a codebook of 200 acoustic labels. This experiment was performed to simulate what is currently the standard IBM recognizer. After training with the forward-backward algorithm, a performance of 79.0 percent correct was obtained.

It was argued in the previous chapters that MMIE will not counter deficiencies in discrete output distributions, since discrete distributions do not make any assumptions which can be false. However, there is still the possibility that MMIE will perform better than MLE due to other invalid assumptions. In fact, in a previous 2000-word vocabulary experiment, Lalit Bahl, Peter deSousa, Robert Mercer and I did find that MMIE can lead to a performance improvement with discrete output distributions [Bahl 86]. In that experiment, as in these experiments with the *e*-set, we did not attempt to correct for an inaccurate language model. Therefore, the performance improvement must have been caused by acoustic-modeling inaccuracies. In particular, it is an important but difficult job to construct accurate word models for a large vocabulary speech recognizer. This is especially true when distributions are shared across many different words, because it is difficult to be sure that a sound in one word is really the same as a sound in another word. It is very possible that it was the use of inaccurate word models which prevented MLE from performing as well as MMIE in this earlier experiment.

Although in the *e*-set experiments, the vowel phones, E and TrE, and the noise phone were shared across all the word models, each word model, with the exception of the model for *d*, contained at least one phone which appeared in no other model. The pronunciation of such a phone was determined solely by instances of the single word in which it occurred. Overall, then, the problem of tying distributions accurately was far less severe for the word models used in the *e*-set experiments than for the word models used in the 2000 word experiment. This may explain why when MMIE was used to estimate the discrete output distributions for the *e*-set in Experiment 2, the performance actually deteriorated slightly.

Having completed these two discrete-parameter experiments, the labeling section of



the signal-processing algorithm was dispensed with, and a series of continuous-parameter experiments was conducted. The performance using the original 20 parameters, after having been projected onto principal components, with diagonal Gaussian output distributions was 65.1 percent, and with full-covariance Gaussians was 76.9 percent. It appears, then, that even after having projected the parameter vectors onto principal components, the off-diagonal terms are important, which can only be the case if the different output distributions have significantly different covariance matrices. This is further corroborated by the fact that the log likelihood of the test data was greater in the full-covariance case than in the diagonal case. If the full-covariance Gaussian had not been a significantly better model for this data, then one would have expected the diagonal model, which has substantially fewer parameters, to more accurately predict test data.

In both these experiments, the system performed worse than in the benchmark discrete experiment, which is consistent with the results of all previous attempts to use continuous parameters in the IBM system. In Chapter 5, we argued that this must be due to the Gaussian model being inaccurate, and it was suggested that MMIE might alleviate this problem. The results of Experiments 5 and 6 demonstrate that this is indeed the case. Although the diagonal Gaussian performance of 74.7 percent is still not up to the discrete performance, the full-covariance Gaussian performance of 83.4 is an improvement over the discrete result.

Having obtained this improvement, the possibility of incorporating time-derivative information by adjoining adjacent parameter vectors, as prescribed in Chapter 7, was investigated in Experiments 7, 8, 9, and 10. In these experiments, parameter vectors from two adjacent frames were combined into one vector of 40 parameters. Notice that for both diagonal Gaussian and full-covariance Gaussian output distributions, the performance with the 40-parameter vectors was worse than the corresponding performance with the original single-frame parameters. However when MMIE was used the situation was reversed, and the adjoined parameters outperformed the single-frame parameters. Here, we see that MMIE was able to make use of additional information that MLE was not able to make use of, presumably because of inaccuracies in the Gaussian or diagonal Gaussian assumptions.

Notice that in both of these experiments the test-data estimate of the average mutual information between word and acoustic signal was extremely low, when compared with the estimates of this quantity made in the other experiments. In both cases, this was caused by a few utterances which had an extremely small probability of being generated by the correct word models. In each of these utterances a couple of frames were outliers with

respect to the appropriate output distributions. It therefore seemed reasonable to replace the Gaussian output distributions with the mixtures of Gaussians that have been proposed by Alan Richter. In Experiment 11, output distributions were used which were mixtures of four Gaussians with variances which were 1, 2, 4, and 8 times variance of the tightest Gaussian in the mixture. I shall refer to such mixtures as an R1248 distributions. The accuracy improved from the ordinary Gaussian result of 64.4 percent to 82.5 percent, which very strongly indicates that there was in fact a problem with outliers.

There is another important observation to be made in the 40-parameter results of experiments 9 and 10. Notice that although the full-covariance Gaussian result with the single-frame MMIE experiment is substantially better than the corresponding diagonal Gaussian result, with the adjoined data the MMIE results for the two distributions are not significantly different. This suggests that there may be too many parameters in the 40-parameter full-covariance distributions. It may be that the data is still more accurately modeled with full-covariance Gaussians, but there may not be enough training data to take advantage of the increased potential accuracy provided by the additional parameters.

In Chapter 6, we described two slightly different antidotes to this problem: projection onto principal components, and projection onto linear discriminants. In experiments 12 through 25, principal components were used to reduce from 40 to 30, and then from 30 to 4 in steps of 2. In Experiments 26 through 39 the same reductions were made but using linear discriminants. In both series of experiments, MLE was used with R1248 output distributions. R1248 distributions were used on the theory that if we are trying to find the best number of parameters for our particular situation, we ought to use models which are likely to be as accurate as possible. The results of these experiments appear in Figures 11.3 and 11.4.

Notice that with fewer than 24 parameters, linear discriminants always perform better than principal components. With more than 24 parameters, they have roughly the same performance. In Figure 11.4, we can see that the test-data estimate of the average mutual information between a word and the corresponding acoustic signal is always greater when linear discriminants are used. In a few experiments, this estimate was negative because it was computed on test data in which there were a few words which were models much better by incorrect models than by correct models. The minimum number of errors as well as the maximum information estimate for principal components occurs with 26 parameters, and with 22 parameters for linear discriminants. This data is, therefore, strong evidence that more information is preserved in the top  $n$  linear discriminants than in the top  $n$  principal

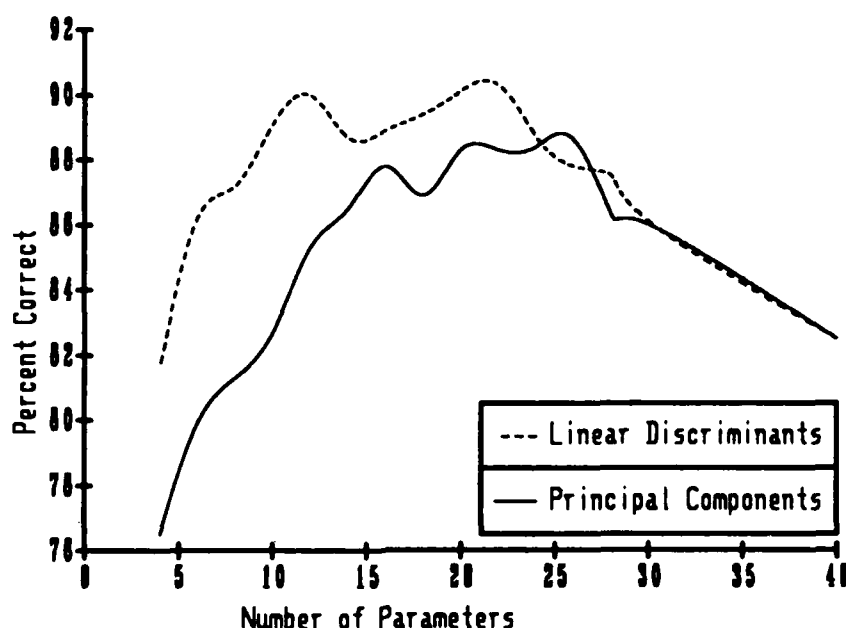


Figure 11.3: Number of Parameters vs. Performance on Test Data

components.

For both principal components and linear discriminants, the performance initially improved as the dimension of the acoustic observation vectors was decreased. Since we cannot have increased the amount of information in these vectors, this improvement must have been due to a reduction in the number of parameters in the output distributions for these dimension-reduced vectors. As the dimension of these observation vectors was further reduced, and more and more information was discarded, the performance began to deteriorate. So we see that there is an important trade-off between using high-dimensional parameter vectors to provide important acoustic information, and using low-dimensional parameter vectors for which probability densities can be accurately estimated.

Since the number of parameters in an output distribution is dependent on the type of distribution, a series of dimension-reduction experiments should have been run for each type of output distribution that we are considering. Due to limited computational resources, this was not done. Rather, a series of experiments was performed with different types of output distributions using the 22-dimensional linear-discriminant vectors that performed optimally when using MLE-estimated R1248 output distributions. The results of these experiments,

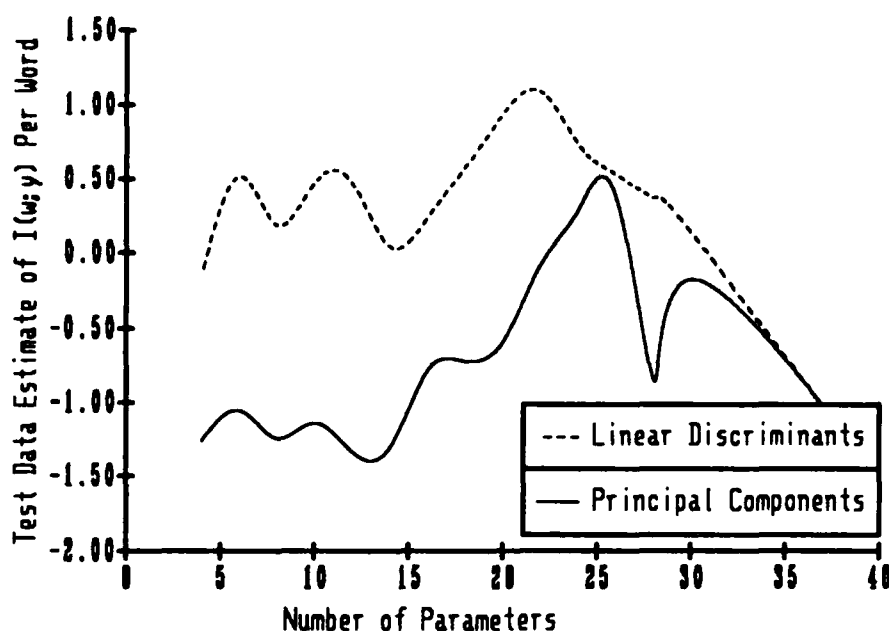


Figure 11.4: Number of Parameters vs. Mutual Information on Test Data

<i>Model Type</i>	<i>MLE Performance</i>	<i>MMIE Performance</i>
Diagonal Gaussian	71.2%	83.1%
Full-Covariance Gaussian	89.1%	91.0%
R12484	90.3%	92.0%
Conditional Gaussian	73.1%	81.8%

Table 11.5: 22-Parameter Results

numbers 40 through 46, are summarized in Table 11.5.

For both MLE and MMIE, the Richter models outperformed the full-covariance Gaussians, which outperformed the diagonal Gaussians. MMIE estimates always outperformed MLE estimates. Furthermore, as one should expect, MMIE resulted in larger improvements for the poorer models. The MMIE R1248 result of 92.0 percent is the best result obtained on this data. It represents a 62 percent decrease in the original discrete error rate.

The last row in Table 11.5 contains results for a conditional hidden Markov model with Gaussian output distribution, as specified in equation (8.2) and in Appendix 13.5. These results, which are substantially worse than those obtained with the unconditional Gaussian

distributions are very disappointing. One possible reason for this poor performance might be that densities are being badly estimated in the conditional model because it has twice as many free parameters as does the normal Gaussian. To test this hypothesis a set of four experiments, numbers 47 through 50, were run with observation vectors which had been projected onto 12 linear discriminants. 12 parameters were chosen since the 12-parameter R1248 MLE result is nearly as good as the 22 parameter result, and since the 12-parameter conditional Gaussian distribution has less parameters than a 22-parameter ordinary Gaussian distribution. The results of these experiments are summarized in Table 11.6. Although the 12-parameter conditional results were significantly better than the 22-parameter results, they were still much worse the unconditional results.

<i>Model Type</i>	<i>MLE Performance</i>	<i>MMIE Performance</i>
Full-Covariance Gaussian	88.7%	90.7%
Conditional Gaussian	84.1%	87.1%

Table 11.6: 12-Parameter Results

Another possible explanation for the poor conditional results is that the unconditional models might have already accounted adequately for the frame-to-frame correlation of the observation vectors. But this cannot have been the case, since, as we can see in Table 11.7, the probability of generating the test data with a conditional model was very much greater than the probability of generating it with an unconditional model. Clearly, there was a severe correlation problem.

In Chapter 8, we argued that it is particularly in steady-state regions of speech that the output-independence assumption is invalid. In the words *b*, *d*, *e*, *g*, *p* and *t*, the only steady-state regions are in the vowels and noise. Because the vowels are the same across all words in the *e*-set, it will not make any difference to the performance on this task if vowels are modeled more accurately. Furthermore if we use a model which does not contribute to the power of the recogniser to discriminate between words, but which has twice as many free parameters as a competing model, we would expect the performance of the recogniser to deteriorate, which it does. It may be, therefore, that the *e*-set is simply not a good task on which to evaluate these type of conditional models.

It may also be that there are other problems with these conditional models in addition to having a lot of free parameters. The assumption in equation (8.2) on which the conditional models used here were based is an approximation of equation (8.1); perhaps it is simply a bad

one. It may be that it is important to know not only what the previous observation vector is, but also the identity of the distribution from which it was generated. It may simply be a bad approximation to model a mixture of distributions which are indexed by the identities of previous distributions, as a single Gaussian. A better approximation may be to assume that such a mixture is a Richter mixture of Gaussians. This would have the advantage of being significantly more robust to outliers that are caused by the single Gaussian being a bad approximation. Since, as we noted above, we clearly have a correlation problem, and since the simple type of conditional models used here do not solve this problem, it will be important in future work to experiment with other types of conditional models, and it seems reasonable to begin with a Richter-model version of equation (8.2).

There was one final experiment conducted on this *e-set* data, a recognition experiment by human listeners. Four hundred words were selected at random from the test data, and listened to by four members of the IBM research staff. On average, they recognized 97.2 percent of the words correctly. When recognition decisions were made by a vote from the four listeners, and each tie was counted as half an error, the performance became 98.0 percent. The conclusion to be drawn from this experiment is that although we have improved the machine performance on this *e-set* task from 79 to 92 percent, there is a long way to go.

<i>Symbol</i>	<i>Interpretation</i>
DIS	Discrete Output Distributions
DG	Diagonal Gaussian Output Distributions
G	Full-Covariance Gaussian Output Distributions
R	R1248 Richter-Type Gaussian Distributions
CG	Conditional Gaussian Output Distributions
PC#	# Principal Components
LD#	# Linear Discriminants
S	Observation Vectors from Single Frames
A	Observation Vectors from Pairs of Adjoined Frames
ML	Maximum Likelihood Estimation
MMI	Maximum Mutual Information Estimation

Table 11.7: Abbreviations used in Table 11.7

#	Type	Results on Training Data			Results on Test Data		
		$\log_2 \text{Pr}(y w)$ Per Frame	$I(w; y)$ Per Word	Percent Correct	$\log_2 \text{Pr}(y w)$ Per Frame	$I(w; y)$ Per Word	Percent Correct
1	DIS-ML	-4.08E+0	2.35E+0	90.0	-4.16E+0	8.53E-1	79.0
2	DIS-MMI	-4.13E+0	2.96E+0	96.1	-4.23E+0	1.06E+0	77.9
3	DG-S-PC20-ML	-1.99E+1	-6.52E+0	69.6	-2.00E+1	-9.80E+0	65.1
4	G-S-PC20-ML	-1.85E+1	8.67E-1	90.4	-1.88E+1	-2.25E+0	76.9
5	DG-S-PC20-MMI	-2.06E+1	1.09E+0	79.8	-2.07E+1	-9.01E-1	74.7
6	G-S-PC20-MMI	-1.88E+1	3.13E+0	99.2	-1.91E+1	6.09E-3	83.4
7	DG-A-PC40-ML	-4.38E+1	-2.10E+1	65.5	-4.39E+1	-2.52E+1	59.8
8	G-A-PC40-ML	-4.15E+1	-6.73E+0	82.2	-4.21E+1	-2.10E+1	64.4
9	DG-A-PC40-MMI	-4.40E+1	2.01E+0	92.1	-4.41E+1	4.43E-1	86.6
10	G-A-PC40-MMI	-4.18E+1	3.17E+0	99.9	-4.23E+1	3.26E-1	86.0
11	R-A-PC40-ML	-4.03E+1	2.32E+0	96.7	-4.09E+1	-1.53E+0	82.5
12	R-A-PC30-ML	-2.82E+1	2.16E+0	96.1	-2.86E+1	-1.74E-1	86.0
13	R-A-PC28-ML	-2.56E+1	1.61E+0	95.7	-2.60E+1	-8.52E-1	86.3
14	R-A-PC26-ML	-2.30E+1	1.85E+0	96.7	-2.34E+1	3.67E-1	88.6
15	R-A-PC24-ML	-2.03E+1	1.68E+0	95.5	-2.60E+1	2.87E-1	88.4
16	R-A-PC22-ML	-1.78E+1	1.73E+0	95.2	-1.82E+1	-7.33E-2	88.3
17	R-A-PC20-ML	-1.50E+1	1.69E+0	94.9	-1.54E+1	-5.90E-1	88.3
18	R-A-PC18-ML	-1.27E+1	1.50E+0	93.4	-1.30E+1	-7.17E-1	86.9
19	R-A-PC16-ML	-1.07E+1	1.44E+0	93.3	-1.10E+1	-7.79E-1	87.8
20	R-A-PC14-ML	-8.52E+0	1.11E+0	91.4	-8.78E+0	-1.31E+0	86.5
21	R-A-PC12-ML	-7.01E+0	1.23E+0	90.5	-7.23E+0	-1.33E+0	85.3
22	R-A-PC10-ML	-5.42E+0	1.13E+0	88.8	-5.63E+0	-1.14E+0	82.7
23	R-A-PC8-ML	-3.54E+0	1.28E+0	86.5	-3.72E+0	-1.24E+0	81.3
24	R-A-PC6-ML	-2.16E+0	1.01E+0	84.2	-2.31E+0	-1.05E+0	79.9
25	R-A-PC4-ML	-1.02E+0	1.58E-1	80.7	-1.10E+0	-1.26E+0	76.5
26	R-A-LD30-ML	-2.64E+1	2.20E+0	96.4	-2.68E+1	1.36E-1	86.1
27	R-A-LD28-ML	-2.38E+1	2.11E+0	94.6	-2.42E+1	3.71E-1	87.5
28	R-A-LD26-ML	-2.13E+1	2.08E+0	95.7	-2.17E+1	5.27E-1	87.8
29	R-A-LD24-ML	-1.90E+1	1.99E+0	95.0	-1.94E+1	7.26E-1	88.7
30	R-A-LD22-ML	-1.75E+1	2.32E+0	95.6	-1.78E+1	1.09E+0	90.3
31	R-A-LD20-ML	-1.57E+1	2.11E+0	94.6	-1.60E+1	9.40E-1	90.1
32	R-A-LD18-ML	-1.37E+1	1.78E+0	94.1	-1.39E+1	5.76E-1	89.4
33	R-A-LD16-ML	-1.19E+1	1.64E+0	93.4	-1.21E+1	2.40E-1	88.9
34	R-A-LD14-ML	-9.96E+0	1.34E+0	92.6	-1.02E+1	4.36E-2	88.7
35	R-A-LD12-ML	-7.87E+0	1.50E+0	92.1	-8.06E+0	4.66E-1	90.0
36	R-A-LD10-ML	-6.05E+0	1.49E+0	91.1	-6.21E+0	4.74E-1	89.1
37	R-A-LD8-ML	-4.59E+0	-1.09E+0	86.9	-4.59E+0	1.89E-1	87.2
38	R-A-LD6-ML	-2.74E+0	1.24E+0	88.8	-2.84E+0	5.15E-1	86.2
39	R-A-LD4-ML	-1.35E+0	7.72E-1	83.9	-1.44E+0	-1.40E-1	81.6
40	DG-A-LD22-ML	-2.03E+1	-3.74E+0	77.4	-2.05E+1	-5.45E+0	71.2
41	DG-A-LD22-MMI	-2.11E+1	1.68E+0	87.7	-2.13E+1	6.24E-2	83.1
42	G-A-LD22-ML	-1.80E+1	3.06E-2	95.1	-1.84E+1	7.38E-1	89.1
43	G-A-LD22-MMI	-1.83E+1	3.17E+0	99.9	-1.86E+1	1.27E+0	91.0
44	R-A-LD22-MMI	-1.76E+1	3.17E+0	99.9	-1.79E+1	1.76E+0	92.0
45	CG-A-LD22-ML	-3.54E+0	2.35E+0	96.0	-4.00E+0	-3.66E+0	73.1
46	CG-A-LD22-MMI	-3.66E+0	3.17E+0	99.9	-4.11E+0	-6.58E-1	81.8
47	G-A-LD12-ML	-8.22E+0	9.15E-1	92.2	-8.45E+0	2.85E-1	88.7
48	G-A-LD12-MMI	-8.63E+0	2.83E+0	97.6	-8.80E+0	7.32E-1	90.7
49	CG-A-LD12-ML	-1.27E+0	2.11E+0	93.5	-1.48E+0	7.01E-1	84.1
50	CG-A-LD12-MMI	-1.37E+0	3.16E+0	99.9	-1.57E+0	1.41E+0	87.1

Table 11.8: E-Set Results (See previous page for symbol definitions.)

## 12. Summary and Conclusion

This thesis has been concerned with the problem of designing a speech-recognition system in such a way that it can extract as much information as possible from a speech waveform about the corresponding word sequence. This extraction process has been broken down into two steps: a signal-processing step which converts a speech waveform into a sequence of information-bearing feature vectors, and a step which models such a sequence. On the one hand, we have seen that we would like to avoid discarding important acoustic information by using a signal-processing step which does as little data reduction as possible. On the other hand, we have seen that we would like the sequence of feature vectors produced by the signal-processing step to be as simple as possible so that it can be accurately modeled by the modeling step. This trade-off has been pivotal to our understanding of the acoustic-modeling problem. It is not enough to simply make sure that a lot of information is contained in the acoustic-feature sequence produced by the signal processor; one must also make sure that this output sequence can be modeled accurately.

We have seen two different ways that acoustic models can be inaccurate: 1) they may be based on invalid assumptions, and 2) they may have so many free parameters that estimation errors will result in errors in the calculation of probabilities or probability densities. In Chapter 2, we argued that our knowledge about speech is at such a primitive stage that if we are not to be completely devastated by the problem of having too many free parameters, then any model of an informative acoustic observation sequence will have to be based on some invalid assumptions. This led us to an investigation of an alternative to MLE, MMIE, which does not derive its *raison d'être* from an implicit assumption of model correctness.

In later chapters, we found that this method of parameter estimation is particularly useful when modeling acoustic feature vectors which lie in a continuous space, such as  $R^N$ . We began our exploration of continuous parameters in Chapter 5, in which it was suggested that there may be important information lost in the common practice of converting from continuous to discrete parameters. We argued that the potential exists to extract more information from the acoustic signal by directly modeling the raw continuous feature vectors. Three different classes of continuous-parameter output distributions were discussed, diagonal Gaussians, full-covariance Gaussians, and Richter-type mixtures of Gaussians. In our *c-set* experiments we found that models based on diagonal Gaussians performed significantly worse than those based on full-covariance Gaussians. For the particular data used in these experiments, then, it was worthwhile to suffer the inaccuracies resulting from estimates of



the additional parameters in full-covariance Gaussians in order to more accurately model the covariance matrices of the acoustic observation vectors. We also found that Richter-type mixtures of Gaussians outperformed single full-covariance Gaussians. This was not surprising since the two types of distributions have roughly the same number of parameters, yet the Richter-models are much more robust to outliers. Our MMIE results indicate, however, that there are still problems remaining with all of these classes of output distributions. In every comparison between MLE and MMIE in a continuous-parameter experiment, the system performed significantly better with MMIE estimates than with MLE estimates.

One problem with modeling continuous feature vectors, is that the number of parameters in the models normally grows with the dimension of the feature vectors. In order to pack as much information as possible into the acoustic feature sequence we would like to use high dimensional feature vectors, which may include a wide variety of sources of acoustic information. But as we increase the dimension of our vectors, we become more vulnerable to estimation errors. In Chapter 6, we described two methods of projecting observation vectors onto lower dimensional planes so as to extract a smaller number of salient features. Our hope was to preserve most of the information in an observation vector while at the same time reducing its dimension. We argued that one should expect the linear-discriminant method to do a better job of preserving acoustic information than the *traditional method* of principal components. In particular, we noted that linear discriminants make use of the classes that are to be discriminated from one another, whereas principal components do not. In the set of experiments performed on the *e-set*, we found that, in fact, as long as we reduced to a small enough dimension to make a difference, better performance was obtained by projecting onto linear discriminants, than by projecting onto principal components.

With this parameter-reduction technique at our disposal, we were equipped to cope with higher dimensional feature vectors. In Chapter 7, we discussed the possibility of including in our feature vectors information on how the speech spectrum changes with time. We saw that this could be done simply by combining adjacent vectors into one vector of twice the dimension. In the *e-set* experiments, we found that including such information substantially improved system performance. Our best results were obtained by first projecting the 40-dimensional vector onto a plane spanned by a smaller number of linear discriminants.

The success obtained from the continuous-parameter experiments, and in particular from those continuous-parameter experiments involving adjoined frames, is attributable to having packed more information into the feature vectors produced by the signal processor. Clearly, there are all kinds of features that can be added to these vectors. One promising

approach to future research would be to develop a wide variety of feature detectors and combine their outputs into high-dimensional feature vectors. Sequences of these vectors could then be modeled in the same way that the adjoined spectral frames were modeled in the  $\epsilon$ -set experiments: the dimension of these vectors could be decreased by projecting them onto linear discriminants, and output distributions could be trained with MMIE in an effort to cope with the problem of not knowing the distribution of vectors of combinations of diverse features.

In Chapter 8, we turned our attention away from the problem of how to create more informative acoustic observation sequences, to one of the fundamental assumptions that is made about these sequences in the hidden-Markov approach, the output independence assumption. We found that in most situations this assumption is likely to be wildly inaccurate. Minor improvements in system performance were obtained in the discrete case by attempting to model the frame-to-frame correlation that is not already accounted for by hidden Markov models which are based on the output-independence assumption. In the continuous-parameter case, however, system performance deteriorated when we attempted to model this correlation. But because the likelihood of generating the test data was significantly greater when correlation models were used, we must assume that there is still a serious modeling problem. Rather than being discouraged by the poor performance we obtained in the continuous-parameter case, we should be enticed by the room for improvement that is likely to be associated with such a serious problem.

In Chapter 9, we again focused our attention on the prospect of increasing the amount of information in the observation sequences output by the signal processor. In this chapter we reasoned that if we do no signal processing at all (save A/D conversion) we can be sure that the signal processor will not be discarding any information. In particular, since human performance on the D/A converted digital signal is far superior than current machine performance, we can be sure that the information necessary to obtain a substantial performance improvement is contained in the digital waveform. While this is very attractive, the trouble with this approach is that now we have to model the speech signal directly, and that is a tough problem. In Chapter 9, we described one such modeling attempt based on linear prediction. The result was very disappointing. Not only did it require tremendous computational resources to use the linear-predictive waveform models discussed in this chapter, but in an experiment on a very small task, these models performed terribly. In spite of this disappointing result, we cannot turn our backs on the fact that if we throw away information about the correct word sequence, system performance will suffer. Although the

waveform-model experiment was a failure, there is no doubt that future improvements in speech recognition will result from modeling acoustic feature sequences that contain more information about the corresponding word sequences.

Our attempt to directly model the waveform involved abandoning the many man years of work that went into the development of the signal-processing algorithm in the IBM system. Perhaps a better approach would be to view the current system as inducing an implicit model on the waveform. We could then view the parameters inside the signal processor as parameters of a waveform model. By estimating first some and then more of these parameters with MMIE, we could hope to gradually extract more and more information from speech waveforms. In this way, we might be able to make use of the extensive research that has gone into the development of feature extraction algorithms, and at the same time benefit from the power of the statistical techniques described in this thesis. †

This thesis has been an investigation into the many facets of the acoustic-modeling problem from a common information-theoretic perspective. What is important at this point in the infancy of automatic speech recognition is not the techniques that have been presented here, but rather an understanding of the problem, from which we can hope that other more sophisticated techniques will evolve.

---

† This approach is, in fact, currently being explored by David Nahamoo and Arthur Nadas, who are attempting to use the programs developed for this thesis to estimate the filter coefficients used by the IBM signal processor.

## 13. Appendices

### 13.1 Differentiation Rules

This section contains a list of simple rules that will be useful in differentiating the probability densities in the sections which follow.

Let  $W$  be a symmetric matrix,  $W' = W$ .

Let  $W_{ij}$  be the element in the  $i$ -th row and  $j$ -th column of  $W$ .

Let  $|W|$  be the determinant of a  $W$ .

Let  $A$  be a matrix.

Let  $u$  and  $v$  be vectors.

Let  $\delta_{ij}$  be 1 if  $i = j$ , and 0 otherwise.

Let  $u \otimes v$  be the outer product of two vectors  $u$  and  $v$ .

1.  $f(W) = |W|$ . What is  $\frac{\partial f}{\partial W}$ ?

$|W| = \sum_i \text{cof}_{ii} W_{ii}$ , where  $\text{cof}_{ij}$  is the cofactor of  $W_{ij}$  in  $W$ .

$$(W^{-1})_{ij} = \frac{\text{cof}_{ji}}{|W|},$$

$$(W^{-1})_{ij} = \frac{\text{cof}_{ij}}{|W|}, \text{ since } W \text{ is symmetric.}$$

$$\frac{\partial f}{\partial W_{ij}} = \text{cof}_{ij}$$

$$= |W| (W^{-1})_{ij}$$

$$\frac{\partial f}{\partial W} = |W| W^{-1}$$

2.  $f(W) = \log |W|$ . What is  $\frac{\partial f}{\partial W}$ ?

$$\begin{aligned} \frac{\partial f}{\partial W} &= \frac{\frac{\partial |W|}{\partial W}}{|W|} \\ &= W^{-1} \end{aligned}$$

3. Let  $g(x)$  be a vector valued function of the real variable  $x$ . Let  $f(x) = g(x)'Wg(x)$ .

What is  $\frac{\partial f}{\partial x}$ ?

$$\begin{aligned} f(x) &= \sum_{ij} g_i(x)W_{ij}g_j(x) \\ \frac{\partial f}{\partial x} &= \sum_{ij} \left( \frac{\partial g_i(x)}{\partial x} W_{ij}g_j(x) + g_i(x)W_{ij} \frac{\partial g_j(x)}{\partial x} \right) \\ &= 2 \sum_{ij} \frac{\partial g_i(x)}{\partial x} W_{ij}g_j(x) \end{aligned}$$

4.  $f(v) = v'Wv$ . What is  $\frac{\partial f}{\partial v}$ ?

$$\begin{aligned} \frac{\partial f}{\partial v_k} &= 2 \sum_{ij} \frac{\partial v_i}{\partial v_k} W_{ij}v_j \\ &= 2 \sum_{ij} \delta_{i,k} W_{ij}v_j \\ &= 2 \sum_j W_{kj}v_j \\ \frac{\partial f}{\partial v} &= 2Wv \end{aligned}$$

5.  $f(v) = (Av + u)'W(Av + u)$ . What is  $\frac{\partial f}{\partial v}$ ?

$$\begin{aligned} \frac{\partial f}{\partial v_k} &= 2 \sum_{ij} \frac{\partial (Av + u)_i}{\partial v_k} W_{ij} (Av + u)_j \\ &= 2 \sum_{ij} \frac{\partial (Av)_i}{\partial v_k} W_{ij} (Av + u)_j \\ &= 2 \sum_{ij} \frac{\partial \sum_l A_{il}v_l}{\partial v_k} W_{ij} (Av + u)_j \\ &= 2 \sum_{ij} A_{ik} W_{ij} (Av + u)_j \\ &= 2 \sum_{ij} A'_{ki} W_{ij} (Av + u)_j \\ \frac{\partial f}{\partial v} &= 2A'W(Av + u) \end{aligned}$$

6.  $f(A) = (A\mathbf{v} + \mathbf{u})' W (A\mathbf{v} + \mathbf{u})$ . What is  $\frac{\partial f}{\partial A}$ ?

$$\begin{aligned} \frac{\partial f}{\partial A_{mn}} &= 2 \sum_{ij} \frac{\partial (A\mathbf{v} + \mathbf{u})_i}{\partial A_{mn}} W_{ij} (A\mathbf{v} + \mathbf{u})_j \\ &= 2 \sum_{ijk} v_k W_{ij} (A\mathbf{v} + \mathbf{u})_j \delta_{im} \delta_{kn} \\ &= 2 \sum_j v_n W_{mj} (A\mathbf{v} + \mathbf{u})_j \\ &= 2 (W (A\mathbf{v} + \mathbf{u}))_m v_n \\ \frac{\partial f}{\partial A} &= 2W (A\mathbf{v} + \mathbf{u}) \otimes \mathbf{v} \end{aligned}$$

7.  $f(W) = \mathbf{u}' W \mathbf{v}$ . What is  $\frac{\partial f}{\partial W}$ ?

$$\begin{aligned} f &= \sum_{ij} u_i W_{ij} v_j \\ \frac{\partial f}{\partial W_{ij}} &= u_i v_j \\ \frac{\partial f}{\partial W} &= \mathbf{u} \otimes \mathbf{v} \end{aligned}$$

### 13.2 Diagonal Gaussian Formulas

Let  $f(\mathbf{x} : \mu, \mathbf{w})$  be the density of an  $n$ -dimensional diagonal Gaussian distribution with mean vector,  $\mu$ , and inverse variance vector,  $\mathbf{w}$ .

**Density:**

$$f(\mathbf{x} : \mu, \mathbf{w}) = \prod_{i=1}^n \left( \frac{w_i}{2\pi} \right)^{1/2} e^{-\frac{1}{2} w_i (x_i - \mu_i)^2}$$

**Derivatives for MMIE:**

$$\frac{\partial f}{\partial \mu_i} = f(\mathbf{x}) w_i (x_i - \mu_i)$$

$$\frac{\partial f}{\partial w_i} = \frac{1}{2} f(\mathbf{x}) \left( \frac{1}{w_i} - (x_i - \mu_i)^2 \right)$$

**Reestimates for MLE:**

Let  $\gamma(t)$  be the probability that the distribution parameterised by  $\mu$  and  $\mathbf{w}$  was used to generate  $y_t$  at time  $t$  given that sample  $\mathbf{y}_1^T$  was observed.

$$\hat{\mu} = \frac{\sum_{t=1}^T \gamma(t) y_t}{\sum_{t=1}^T \gamma(t)}$$

$$\hat{w}_i = \frac{\sum_{t=1}^T \gamma(t)}{\sum_{t=1}^T \gamma(t) (x_{t,i} - \hat{\mu}_i)^2}$$

### 13.3 Full-Covariance Gaussian Formulas

Let  $f(\mathbf{z} : \mu, W)$  be the density of an  $n$ -dimensional Gaussian distribution with mean vector  $\mu$  and inverse covariance matrix  $W$ .

**Density:**

$$f(\mathbf{z} : \mu, W) = \frac{|W|^{1/2}}{(2\pi)^{n/2}} e^{-\frac{1}{2}(\mathbf{z}-\mu)'W(\mathbf{z}-\mu)}$$

**Derivatives for MMIE:**

$$\frac{\partial f}{\partial \mu} = f(\mathbf{z})W(\mathbf{z} - \mu)$$

$$\begin{aligned} \frac{\partial f}{\partial W} &= \frac{1}{2} \frac{1}{(2\pi)^{n/2}} |W|^{-1/2} |W| W^{-1} e^{-\frac{1}{2}(\mathbf{z}-\mu)'W(\mathbf{z}-\mu)} - \frac{1}{2} f(\mathbf{z}) (\mathbf{z} - \mu) \otimes (\mathbf{z} - \mu) \\ &= \frac{1}{2} f(\mathbf{z}) W^{-1} - \frac{1}{2} f(\mathbf{z}) (\mathbf{z} - \mu) \otimes (\mathbf{z} - \mu) \\ &= \frac{1}{2} f(\mathbf{z}) (W^{-1} - (\mathbf{z} - \mu) \otimes (\mathbf{z} - \mu)) \end{aligned}$$

**Reestimates for MLE:**

Let  $\gamma(t)$  be the probability that the distribution parameterised by  $\mu$  and  $W$  was used to generate  $\mathbf{y}_t$  at time  $t$  given that sample  $\mathbf{y}_1^T$  was observed.

$$\hat{\mu} = \frac{\sum_{t=1}^T \gamma(t) \mathbf{y}_t}{\sum_{t=1}^T \gamma(t)}$$

$$\hat{W} = \left[ \frac{\sum_{t=1}^T \gamma(t) (\mathbf{y}_t - \hat{\mu}_t) \otimes (\mathbf{y}_t - \hat{\mu}_t)}{\sum_{t=1}^T \gamma(t)} \right]^{-1}$$



## 13.4 Richter Type Gaussian Formulas

Let  $f(\mathbf{x} : \mu, W, m, c, \lambda)$  be the density of an  $n$ -dimensional mixture of  $m$  Gaussian distributions with common mean  $\mu$ , common inverse covariance matrix  $W$ , vector of variance multiplication factors  $c$ , and vector mixing factors  $\lambda$  [Richter 86].

Density:

$$f_i(\mathbf{x} : \mu, W, c_i) = \frac{|W|^{1/2}}{(2\pi)^{n/2} c_i^{n/2}} e^{-\frac{1}{2c_i}(\mathbf{x}-\mu)'W(\mathbf{x}-\mu)}$$

$$f(\mathbf{x} : \mu, W, m, c, \lambda) = \sum_{i=1}^m \lambda_i f_i(\mathbf{x} : \mu, W, c_i)$$

Derivatives for MMIE:

$$\frac{\partial f}{\partial \lambda_i} = f_i(\mathbf{x})$$

$$\begin{aligned} \frac{\partial f}{\partial \mu} &= \sum_{i=1}^m \frac{1}{c_i} \lambda_i f_i(\mathbf{x}) W (\mathbf{x} - \mu) \\ &= W (\mathbf{x} - \mu) \sum_{i=1}^m \frac{1}{c_i} \lambda_i f_i(\mathbf{x}) \end{aligned}$$

$$\begin{aligned} \frac{\partial f}{\partial W} &= \sum_{i=1}^m \left( \frac{1}{2} W^{-1} \lambda_i f_i(\mathbf{x}) - \frac{1}{2c_i} \lambda_i f_i(\mathbf{x}) (\mathbf{x} - \mu) \otimes (\mathbf{x} - \mu) \right) \\ &= \frac{1}{2} f(\mathbf{x}) W^{-1} - \frac{1}{2} (\mathbf{x} - \mu) \otimes (\mathbf{x} - \mu) \sum_{i=1}^m \frac{1}{c_i} \lambda_i f_i(\mathbf{x}) \end{aligned}$$

$$\begin{aligned} \frac{\partial f}{\partial c_i} &= \frac{-n \lambda_i |W|^{1/2}}{2 (2\pi)^{n/2} c_i^{(n+2)/2}} e^{-\frac{1}{2c_i}(\mathbf{x}-\mu)'W(\mathbf{x}-\mu)} + \frac{\lambda_i f_i(\mathbf{x}) (\mathbf{x} - \mu)' W (\mathbf{x} - \mu)}{2c_i^2} \\ &= \frac{\lambda_i f_i(\mathbf{x})}{2c_i} \left( -n + \frac{(\mathbf{x} - \mu)' W (\mathbf{x} - \mu)}{c_i} \right) \end{aligned}$$

**Reestimates for MLE:**

Let  $\gamma_i(t)$  be the probability that the sub-distribution  $i$  of the mixture parameterized by  $\mu, W, m, c$ , and  $\lambda$  was used to generate  $y_t$  at time  $t$  given that the sample  $y_1^T$  was observed. If  $\mathcal{D}$  is the set of all transition with output distributions that are tied to the distribution we are concerned with, then

$$\gamma_i(t) = \sum_{j \rightarrow k \in \mathcal{D}} \alpha_j(t) a_{jk} \lambda_i f_k(y_t) \beta_k(t).$$

By solving simultaneous equations (one for each parameter  $\phi$ ) of the form

$$\frac{\partial}{\partial \phi} \sum_{t=1}^T \sum_{i=1}^m \gamma_i(t) \log f_i(y_t) = 0, \quad (13.1)$$

while maintaining the constraint

$$\sum_{i=1}^m \lambda_i = 1, \quad (13.2)$$

we obtain the following reestimation formulas.

$$\hat{\lambda}_i = \frac{\sum_{t=1}^T \gamma_i(t)}{\sum_{t=1}^T \sum_{k=1}^m \gamma_k(t)}$$

$$\hat{c}_i = \frac{\sum_{t=1}^T \gamma_i(t) (y_t - \hat{\mu})' \hat{W} (y_t - \hat{\mu})}{n \sum_{t=1}^T \sum_{k=1}^m \gamma_k(t)}$$

In practice, to avoid the absurd solution in which some  $c_i \rightarrow 0$ , we do not reestimate the  $c_i$ 's.

$$\hat{\mu} = \frac{\sum_{k=1}^m \frac{1}{c_k} \sum_{t=1}^T \gamma_k(t) y_t}{\sum_{k=1}^m \frac{1}{c_k} \sum_{t=1}^T \gamma_k(t)}$$

$$\hat{W}^{-1} = \frac{\sum_{t=1}^T \sum_{k=1}^m \frac{\gamma_k(t) y_t \otimes y_t}{c_k} - \hat{\mu} \otimes \hat{\mu} \sum_{t=1}^T \sum_{k=1}^m \frac{\gamma_k(t)}{c_k}}{\sum_{t=1}^T \sum_{k=1}^m \gamma_k(t)}$$

## 13.5 Conditional Gaussian Formulas

Suppose that the sample vector  $\mathbf{z}$  is composed of two  $n$ -dimensional vectors

$$\mathbf{z} = (\mathbf{z}_1, \mathbf{z}_2).$$

Suppose further that  $\mathbf{z}$  is a sample from a multivariate Gaussian, with mean

$$\boldsymbol{\mu} = (\boldsymbol{\mu}_1, \boldsymbol{\mu}_2),$$

covariance matrix

$$\Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma'_{12} & \Sigma_{22} \end{pmatrix},$$

and inverse covariance matrix

$$V = \begin{pmatrix} V_{11} & V_{12} \\ V'_{12} & V_{22} \end{pmatrix}.$$

In this section we will consider the conditional density  $f(\mathbf{z}_2 | \mathbf{z}_1)$ .

**Density:**

Let  $\mathbf{z} = \mathbf{z} - \boldsymbol{\mu}$ ,  $\mathbf{z}_1 = \mathbf{z}_1 - \boldsymbol{\mu}_1$ , and  $\mathbf{z}_2 = \mathbf{z}_2 - \boldsymbol{\mu}_2$ . The assumption that  $f(\mathbf{z})$  is a Gaussian density is that

$$f(\mathbf{z}_1, \mathbf{z}_2) = \frac{|V|^{1/2}}{(2\pi)^{n/2}} e^{-\frac{1}{2} \mathbf{z}' V \mathbf{z}},$$

where

$$\mathbf{z}' V \mathbf{z} = \mathbf{z}'_1 V_{11} \mathbf{z}_1 + 2 \mathbf{z}'_1 V_{12} \mathbf{z}_2 + \mathbf{z}'_2 V_{22} \mathbf{z}_2.$$

By the definition of conditional probability,

$$\begin{aligned} f(\mathbf{z}_2 | \mathbf{z}_1) &= \frac{f(\mathbf{z}_1, \mathbf{z}_2)}{\int_{\mathbf{z}_2} f(\mathbf{z}_1, \mathbf{z}_2)} \\ &= \frac{e^{-\frac{1}{2} \mathbf{z}' V \mathbf{z}}}{\int_{\mathbf{z}_2} e^{-\frac{1}{2} \mathbf{z}' V \mathbf{z}}} \\ &= \frac{e^{-\frac{1}{2} (\mathbf{z}'_2 V_{22} \mathbf{z}_2 + 2 \mathbf{z}'_1 V_{12} \mathbf{z}_2)}}{\int_{\mathbf{z}_2} e^{-\frac{1}{2} (\mathbf{z}'_2 V_{22} \mathbf{z}_2 + 2 \mathbf{z}'_1 V_{12} \mathbf{z}_2)}}. \end{aligned}$$

By completing the square we find that

$$\mathbf{z}'_2 V_{22} \mathbf{z}_2 + 2 \mathbf{z}'_1 V_{12} \mathbf{z}_2 = \left( \mathbf{z}_2 + V_{22}^{-1} V_{21} \mathbf{z}_1 \right)' V_{22} \left( \mathbf{z}_2 + V_{22}^{-1} V_{21} \mathbf{z}_1 \right) - \mathbf{z}'_1 V_{12} V_{22}^{-1} V_{21} \mathbf{z}_1,$$

and

$$\int_{\mathbf{z}_2} e^{-\frac{1}{2}(\mathbf{z}_2' V_{22} \mathbf{z}_2 + 2\mathbf{z}_1' V_{12} \mathbf{z}_2)} = \left( \frac{(2\pi)^{n/2}}{|V_{22}|^{1/2}} \right) e^{\frac{1}{2}\mathbf{z}_1' V_{12} V_{22}^{-1} V_{21} \mathbf{z}_1}.$$

Substituting this in the formula for  $f(\mathbf{z}_2 | \mathbf{z}_1)$  above, we find

$$\begin{aligned} f(\mathbf{z}_2 | \mathbf{z}_1) &= \frac{|V_{22}|^{1/2}}{(2\pi)^{n/2}} e^{-\frac{1}{2}(\mathbf{z}_2' V_{22} \mathbf{z}_2 + 2\mathbf{z}_1' V_{12} \mathbf{z}_2 + \mathbf{z}_1' V_{12} V_{22}^{-1} V_{21} \mathbf{z}_1)} \\ &= \frac{|V_{22}|^{1/2}}{(2\pi)^{n/2}} e^{-\frac{1}{2}(\mathbf{z}_2 + V_{22}^{-1} V_{21} \mathbf{z}_1)' V_{22} (\mathbf{z}_2 + V_{22}^{-1} V_{21} \mathbf{z}_1)} \\ &= \frac{|W|^{1/2}}{(2\pi)^{n/2}} e^{-\frac{1}{2}(\mathbf{z}_2 - (\mu_2 + C(\mathbf{z}_1 - \mu_1)))' W (\mathbf{z}_2 - (\mu_2 + C(\mathbf{z}_1 - \mu_1)))}, \end{aligned}$$

where the inverse covariance matrix  $W = V_{22}$ , and the conditioning matrix  $C = -V_{22}^{-1} V_{21}$ .

From

$$\begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix} \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix} = I,$$

we find that  $W^{-1} = V_{22}^{-1} = \Sigma_{22} - \Sigma_{21} \Sigma_{11}^{-1} \Sigma_{12}$ , and that  $C = -V_{22}^{-1} V_{21} = \Sigma_{21} \Sigma_{11}^{-1}$ . We can conclude therefore that  $\mathbf{z}_2$  has a Gaussian density with mean  $\mu_2 + \Sigma_{21} \Sigma_{11}^{-1} (\mathbf{z}_1 - \mu_1)$ , and covariance  $\Sigma_{22} - \Sigma_{21} \Sigma_{11}^{-1} \Sigma_{12}$ .

**Derivatives for MMIE:**

$$\frac{\partial f}{\partial \mu_1} = -f(\mathbf{z}_2 | \mathbf{z}_1) C' W (\mathbf{z}_2 - (\mu_2 + C(\mathbf{z}_1 - \mu_1)))$$

$$\frac{\partial f}{\partial \mu_2} = f(\mathbf{z}_2 | \mathbf{z}_1) W (\mathbf{z}_2 - (\mu_2 + C(\mathbf{z}_1 - \mu_1)))$$

$$\frac{\partial f}{\partial W} = \frac{1}{2} f(\mathbf{z}_2 | \mathbf{z}_1) (W^{-1} - (\mathbf{z}_2 - (\mu_2 + C(\mathbf{z}_1 - \mu_1))) \otimes (\mathbf{z}_2 - (\mu_2 + C(\mathbf{z}_1 - \mu_1))))$$

$$\frac{\partial f}{\partial C} = f(\mathbf{z}_2 | \mathbf{z}_1) W (\mathbf{z}_2 - (\mu_2 + C(\mathbf{z}_1 - \mu_1))) \otimes (\mathbf{z}_1 - \mu_1)$$

**Reestimates for MLE:**

Let  $\gamma(t)$  be the probability that the distribution parameterized by  $\mu_1, \mu_2, C$ , and  $W$  was used to generate  $\mathbf{x}_2 = \mathbf{y}_t$  with  $\mathbf{x}_1 = \mathbf{y}_{t-1}$  at time  $t$  given that sample  $\mathbf{y}_1^T$  was observed.

$$\hat{\mu}_1 = \frac{\sum_{t=1}^T \gamma(t) \mathbf{y}_{t-1}}{\sum_{t=1}^T \gamma(t)}$$

$$\hat{\mu}_2 = \frac{\sum_{t=1}^T \gamma(t) \mathbf{y}_t}{\sum_{t=1}^T \gamma(t)}$$

$$\hat{C} = \left[ \sum_{t=1}^T \gamma(t) (\mathbf{y}_t - \hat{\mu}_2) \otimes (\mathbf{y}_{t-1} - \hat{\mu}_1) \right] \left[ \sum_{t=1}^T \gamma(t) (\mathbf{y}_{t-1} - \hat{\mu}_1) \otimes (\mathbf{y}_{t-1} - \hat{\mu}_1) \right]^{-1}$$

$$\hat{W}^{-1} = \frac{\sum_{t=1}^T \gamma(t) (\mathbf{y}_t - (\hat{\mu}_2 + \hat{C}(\mathbf{y}_{t-1} - \hat{\mu}_1))) \otimes (\mathbf{y}_t - (\hat{\mu}_2 + \hat{C}(\mathbf{y}_{t-1} - \hat{\mu}_1)))}{\sum_{t=1}^T \gamma(t)}$$

### 13.6 LPC Formulas

Let  $f(y_t | z_{t-n}^{t-1} : a, w)$  be the density of an  $n$ th-order LPC model, with prediction coefficients  $a_1, \dots, a_n$  and inverse error variance  $w$ . In the case of a vanilla LPC model,  $z_{t-i} = y_{t-i}$ , for  $1 \leq i \leq n$ . Suppose we have an LPC model enhanced by a pitch tracker in which  $y_t$  is predicted from  $y_{t-k}, \dots, y_{t-1}$ , and from  $y_{(t-p_t-m)}, \dots, y_{(t-p_t+m)}$ , where  $n = k + 2m + 1$ , and where  $p_t$  is the pitch tracker's estimate of the pitch period at time  $t$ . In this case,  $z_{t-i} = y_{t-i}$ , for  $1 \leq i \leq k$ , and  $z_{(t-k-m-1+i)} = y_{(t-p_t+i)}$ , for  $-m \leq i \leq m$ .

**Density:**

$$\epsilon_t = y_t - \sum_{i=1}^n a_i z_{t-i}$$

$$f(y_t | z_{t-n}^{t-1}) = \left(\frac{w}{2\pi}\right)^{1/2} e^{-\frac{1}{2}w\epsilon_t^2}$$

**Derivatives for MMIE:**

$$\frac{\partial f}{\partial a_i} = f(y_t | z_{t-n}^{t-1}) w z_{t-i} \epsilon_t$$

$$\frac{\partial f}{\partial w} = \frac{1}{2} f(y_t | z_{t-n}^{t-1}) \left(\frac{1}{w} - \epsilon_t^2\right)$$

**Reestimates for MLE:**

Let  $\gamma(t)$  be the probability that the distribution parameterized by  $a$  and  $w$  was used to generate  $y_t$  at time  $t$ , given that the sample  $y_1^T$  was observed. To reestimate the prediction coefficients, solve the following system of  $n$  linear equations:

$$\sum_{t=1}^T \gamma(t) y_t z_{t-i} = \sum_{t=1}^T \gamma(t) z_{t-i} \sum_{j=1}^n a_j z_{t-j}, \quad \text{for } i = 1 \text{ to } n.$$

$w$  can then be reestimated as

$$\hat{w} = \frac{\sum_{t=1}^T \gamma(t)}{\sum_{t=1}^T \gamma(t) \left(y_t - \sum_{i=1}^n \hat{a}_i y_{t-i}\right)^2}.$$

## 14. References

- [Ash 65] R.B. Ash  
*Information Theory*  
Interscience Publishers, New York, 1965.
- [Averbuch 86] A. Averbuch, L. Bahl, R. Bakis, P. Brown, A. Cole, G. Daggett, S. Das, K. Davies, S. De Gennaro, P. de Souza, E. Epstein, D. Fraleigh, F. Jelinek, S. Katz, B. Lewis, R. Mercer, A. Nadas, D. Nahamoo, M. Picheny, G. Shichman and P. Spinelli  
"An IBM PC based large-vocabulary isolated-utterance speech Recognizer"  
*Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Tokyo, pages 53-56, April 1986.
- [Bahl 80] L.R. Bahl, R. Bakis, F. Jelinek and R.L. Mercer  
"Language-model/acoustic channel balance mechanism"  
*IBM Technical Disclosure Bulletin*, Volume 23, Number 7B, pages 3464-3465, December 1980.
- [Bahl 81] L.R. Bahl, R. Bakis, P.S. Cohen, A.G. Cole, F. Jelinek, B.L. Lewis and R.L. Mercer  
"Continuous parameter acoustic processing for recognition of a natural speech corpus"  
*Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Atlanta, pages 1149-1155, April 1981.
- [Bahl 83] L.R. Bahl, F. Jelinek and R.L. Mercer  
"A maximum likelihood approach to continuous speech recognition"  
*IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume PAMI-5, Number 2, pages 179-190, March 1983.
- [Bahl 86] L.R. Bahl, P.F. Brown, P.V. deSouza and R.L. Mercer  
"Maximum mutual information estimation of hidden Markov model parameters for speech recognition"  
*Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Tokyo, pages 49-52, April 1986.
- [Baker 75] J.K. Baker  
"The DRAGON system — an overview,"  
*IEEE Transactions on Acoustics, Speech and Signal Processing*, Volume ASSP-23, pages 24-29, February 1975.
- [Baker 79] J.K. Baker  
"Stochastic modeling for automatic speech understanding"  
*Speech Recognition*, R.A. Reddy ed., Academic Press, New York, 1979.
- [Bakis 76] R. Bakis  
"Continuous speech recognition via centisecond acoustic states"  
presented at the 91st Meeting Acoustical Society of America, Washington, DC, April 1976; also IBM Research Report RC-5971, IBM Research Center, Yorktown Heights, NY, April 1976.

- [Baum 72] L.E. Baum  
"An inequality and associated maximization technique in statistical estimation of probabilistic functions of a Markov process"  
*Inequalities*, Volume 3, pages 1-8, 1972
- [Bellman 57] R. Bellman  
*Dynamic Programming*  
Princeton University Press, Princeton N.J. 1957
- [Bradshaw 82] G.L. Bradshaw, R.A. Cole and Z. Li  
"Comparison of learning techniques in speech recognition"  
*Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Paris, pages 554-557, May 1982.
- [Brown 83] P.F. Brown, C.H. Lee and J.C. Spohrer  
"Bayesian adaptation in Speech recognition"  
*Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Boston, pages 761-764, April 1983.
- [Brown 84] P.F. Brown  
"Speech recognition including biased Principal components"  
U.S. Patent Application Number 593894, filed March 1984.
- [Cohen 87] J.R. Cohen  
"Application of an auditory model to speech recognition"  
forthcoming in the *Journal of the Acoustic Society of America*.
- [Cole 80] R.A. Cole, A.I. Rudnick, V.W. Zue and D.R. Reddy  
"Speech as patterns on paper"  
*Perception and production of fluent speech*, edited by R.A. Cole, Lawrence Erlbaum Associates, Hillsdale, N.J., 1980
- [Cole 86] R.A. Cole, R.M. Stern and M.J. Lasry  
"Performing fine phonetic distinctions: templates vs. features"  
*Invariance and variability of features in spoken English letters*, edited by J. Perkell *et al.*, Lawrence Erlbaum, New York, N.Y., 1986.
- [Dempster 77] A.P. Dempster, N.M. Laird and D.B. Rubin  
"Maximum likelihood from incomplete data via the EM algorithm"  
*Journal of the Royal Statistical Society, Series B* Volume 39, pages 1-38, 1977.
- [Dixon 76] N.R. Dixon and H.F. Silverman  
"A general language-oriented decision Implementation system (GLODIS): Its Application to Continuous-Speech Segmentation"  
*IEEE Transactions on Acoustics, Speech and Signal Processing*, Volume ASSP-24, Number 2, pages 137-162, April 1976.
- [Friedman 67] H.P. Friedman and J. Rudin  
"On some invariant criteria for grouping data"  
*American Statistical Association Journal*, pages 1159-1178, December 1967.
- [Gallager 68] R.G. Gallager  
*Information Theory and Reliable Communication*  
John Wiley and Sons, Inc., New York, 1968.



- [Hinton 86] G.E. Hinton and T.J. Sejnowski  
"Learning and relearning in Boltzmann machines"  
*Parallel Distributed Processing*, Volume 1, edited by D.E. Rumelhart, J.L. McClelland, and the PDP Research Group, M.I.T. Press 1986.
- [Huber 81] P.J. Huber  
*Robust statistics*  
John Wiley and Sons, Inc., New York, 1981.
- [Itakura 75] F. Itakura  
"Minimum prediction residual principle applied to speech recognition"  
*IEEE Transactions on Acoustics, Speech and Signal Processing*, Volume ASSP-23, pages 67-72, February 1975.
- [Jelinek 80] F. Jelinek and R.L. Mercer  
"Interpolated estimation of Markov source parameters from sparse data"  
*Proceedings of the Workshop on Pattern Recognition in Practice*, Amsterdam, The Netherlands: North-Holland, May 1980.
- [Jelinek 85] F. Jelinek  
"Self-organized language modeling for speech recognition"  
unpublished, 1985.
- [Katz 87] S. Katz  
"Estimation of probabilities from sparse data for the language model component of a speech recognizer"  
*IEEE Transactions on Acoustics, Speech and Signal Processing*, Volume ASSP-35, Number 3, pages 400-401, March 1987.
- [Lasry 84] M.J. Lasry and R.M. Stern  
"Unsupervised adaptation to new speakers in feature-based letter recognition"  
*Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, San Diego, pages 17.6.1-17.6.4, April 1984.
- [Lowerre 76] B.T. Lowerre  
"The HARPY speech recognition system"  
Ph.D. Thesis, Carnegie-Mellon University, April 1976
- [Lucassen 82] J.M. Lucassen  
"Discovering Phonemic baseforms: an information theoretic approach"  
IBM Research Report RC 9833, February 8, 1983.
- [Makhoul 75] J. Makhoul  
"Linear prediction: a tutorial review"  
*IEEE Transactions on Communication*, Volume 63 Number 4, pages 561-580, 1975.
- [Makhoul 86] J. Makhoul  
Personal communication  
I.B.M. Watson Research Center in Yorktown Heights, N.Y., 1986.
- [Markel 76] J.D. Markel and A.H. Gray  
*Linear Prediction of Speech*  
Springer, Berlin, 1976

- [Meilijson 85] I. Meilijson  
"A fast improvement to the EM algorithm on its own terms"  
Forthcoming in the *Journal of the Royal Statistical Society, Series B*, 1987.
- [Mercer 86] R.L. Mercer, A. Nadas and D. Nahamoo  
"Maximally informative reduction of the dimension of speech parameters"  
IBM Invention Disclosure Number Y0886-0168, March 1986.
- [Nadas 81] A. Nadas, L.R. Bahl, R. B.S. Cohen, A.G. Cole F. Jelinek and B.L. Lewis  
"Continuous speech recognition with automatically selected acoustic prototypes obtained by either bootstrapping or clustering"  
*Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Atlanta, Georgia, pages 1153-1155, March 1981.
- [Nadas 83] A. Nadas  
"A decision-theoretic formulation of a training problem in speech recognition and a comparison of training by unconditional versus conditional maximum likelihood"  
*IEEE Transactions on Acoustics, Speech and Signal Processing*, Volume ASSP-31, Number 4, pages 814-817, August 1983.
- [Nadas 86] A. Nadas  
Personal communication  
I.B.M. Watson Research Center in Yorktown Heights, N.Y., 1986.
- [Patterson 86] N.J. Patterson  
Personal communication  
I.D.A. Communication Research Division, Princeton, N.J., 1986.
- [Rabiner 79] L.R. Rabiner, S.E. Levinson, A.E. Rosenberg and J.G. Wilpon  
"Speaker-independent recognition of isolated words using clustering techniques"  
*IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-27, Number 4, pages 336-349, August 1979.
- [Rabiner 81] L.R. Rabiner and J.G. Wilpon  
"Isolated word recognition using a two-pass pattern recognition approach"  
*Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Atlanta, Georgia, pages 724-727, April 1981.
- [Poritz 86] A.B. Poritz and A.G. Richter  
"On hidden Markov models in isolated word recognition"  
*Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Tokyo, pages 705-708, April 1986.
- [Richter 86] A.G. Richter  
"Modeling of continuous speech observations"  
Presented at the Advances in Speech Processing Conference, IBM Europe Institute, Oberlech, Austria, July 1986.

- [Shannon 48] C.E. Shannon  
"A mathematical theory of communication"  
*Bell System Technical Journal*, 27, pages 379-423 (Part I), pages 623-656 (Part II), 1948.
- [Stern 83] R.M Stern and M.J. Lasry  
"Dynamic speaker adaptation for isolated letter recognition using MAP estimation"  
*Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Boston, pages 734-737, April 1983.
- [Stern 87] R.M Stern and M.J. Lasry  
"Dynamic speaker adaptation for feature-based isolated-word recognition"  
forthcoming in *IEEE Transactions on Acoustics, Speech and Signal Processing*, June 1987.
- [Swartzlander 75] E.E. Swartzlander, Jr. and A.G. Alexopoulos  
"The sign/Logarithm Number system"  
*IEEE Transactions on Computers*, C-24, pages 1238-1242, 1975.
- [Thomson 85] P.J. Thomson and P. de Sousa  
"Speech recognition using LPC distance measures"  
*Handbook of Statistics*, Volume 5, pages 389-412, edited by E.J. Hannan, P.R. Krishnaiah and M.M. Rao, Elsevier Science Publishers B.V. (North Holland) The Netherlands, 1985.
- [Viterbi 67] A.J. Viterbi  
"Error bounds for convolutional codes and an asymptotically optimum decoding algorithm"  
*IEEE Transactions on Information Theory*, IT-13, pages 260-269, 1967.
- [Waibel 81] A. Waibel and B. Yegnanarayana  
"Comparative study of nonlinear time warping techniques in isolated word speech recognition systems"  
Carnegie-Mellon Computer Science Technical Report  
CMU-CS-81-125.
- [Wilks 61] S.S. Wilks  
*Mathematical Statistics*,  
John Wiley and Sons, Inc. New York, 1961.

END

DATE

FILMD

3-88

DTIC